INFORMATICA

Andrei BRAICOV

TURBO PASCAL

Culegere de probleme

PRUT INTERNATIONAL

INFORMATICA

Andrei BRAICOV

TURBO PASCAL

Culegere de probleme

Ediția a II-a

PRET INTERNATIONAL

Lucrarea a fost aprobată de Comisia metodică şi Senatul Universităţii de Stat din Tiraspol.

Toate drepturile asupra acestei ediţii aparţin Editurii *Prut Internaţional.*
Reproducerea integrală sau parţială a textului sau a ilustraţiilor din această carte
este permisă numai cu acordul scris al editurii.

Autor: *Andrei Braicov*, doctor, conferenţiar universitar, UST

Recenzenţi: *Liubomir Chiriac*, doctor, conferenţiar universitar, UST
Nicolae Objelean, doctor, conferenţiar universitar, USM

Redactor: *Tatiana Rusu*
Corector: *Elena Bivol*
Coperta: *Sergiu Stanciu*
Paginare computerizată: *Alexandru Colibaba*

© Editura *Prut Internaţional,* 2007
© *Andrei Braicov,* 2007

Editura *Prut Internaţional*, str. George Enescu nr. 6, bl. 1, Chişinău MD 2064
Tel.: 74.93.18, 75.18.74; fax: 50.87.20; e-mail: prut@mtc.md

Difuzare: Societatea de Distribuţie a Cărţii *Pro-Noi*
str. Alba-Iulia nr. 23, bl. 1A, Chişinău
Tel.: 51.68.17; www.pronoi.md
e-mail: info@pronoi.md

Imprimat la F.E.P. *Tipografia Centrală.* Comanda nr. 4698

CZU 004.43(076.5)

ISBN 9975-69-788-7

Cuprins

Prefața 3

§1. Tipuri de date simple şi operații de bază asupra lor 4

§2. Structuri ramificate (alternative) 17

§3. Structuri repetitive 30

§4. Tablouri unidimensionale (vectori) 45

§5. Tablouri bidimensionale 58

§6. Şiruri de caractere 74

§7. Numere aleatoare 85

§8. Tipul mulțime (Set) 90

§9. Tipul înregistrare (Record) 96

§10. Tipul fişier 107

§11. Subprograme 121

§12. Subprograme recursive 131

§13. Alocarea dinamică a memoriei (Tipul *referință*).
Structuri dinamice de date 141

§14. Unit-uri proprii 163

§15. Posibilități grafice 170

§16. Programarea orientată pe obiecte 187

§17. Grafuri (neorientate) 198

Anexa 1. Unit-ul CRT 215

Anexa 2. Unit-ul Graph 218

Anexa 3. Unit-ul DOS 227

Bibliografie 231

Fiicei mele, Daniela

PREFAȚĂ

Informatica (în special programarea) este o ştiinţă la care aspiră mulţi, dar pe care o însuşesc cu succes doar cei care exersează permanent. Iscusinţa de a programa se dobîndeşte cu anii, de aceea cu cît mai „devreme" elevul va descoperi algoritmi, cu atît mai mult va avansa şi mai uşor va reuşi să se pătrundă de frumuseţea, farmecul şi aplicabilitatea acestui domeniu.

Odată „prins", gustul de programare sporeşte exponenţial şi-l face pe elev să se avînte în noi căutări. Dar... acest lucru nu se întîmplă imediat. Mai întîi trebuie învăţat „alfabetul programării". Se pare că unul dintre cele mai potrivite limbaje de programare atît pentru instruire cît şi pentru elaborări de program ar fi Turbo Pascal.

Am scris această culegere cu o deosebită plăcere, entuziasmat de „provocările" şi doleanţele studenţilor, elevilor şi profesorilor. Lucrarea se adresează anume lor şi tuturor celor care doresc să programeze în Pascal. Unii dintre algoritmii prezentaţi nu sînt neapărat cei mai raţionali, acest lucru fiind comis din considerente metodice. Cititorul poate căuta alte soluţii (care posibil îi vor părea mai clare).

Problemele rezolvate, precum şi cele propuse au fost verificate pe parcursul mai multor ani „pe pielea" studenţilor Universităţii de Stat din Tiraspol şi a elevilor liceelor în care s-au desfăşurat practicile pedagogice.

Culegerea depăşeşte nivelul prevăzut de curriculumul liceal. Luînd în consideraţie că deseori cuvîntul „opţional" este interpretat de elevi ca „neesenţial", secvenţele considerate în curriculum suplimentare sau opţionale nu au fost puse în evidenţă.

Fiecare capitol al cărţii constă din trei părţi:

- Sugestii teoretice;
- Probleme rezolvate;
- Exerciţii şi probleme propuse, care sînt structurate pe trei niveluri:
  - **A.** Exerciţii şi probleme pentru fixarea şi consolidarea cunoştinţelor;
  - **B.** Probleme pentru antrenament;
  - **C.** Probleme pentru dezvoltare.

Recomand utilizatorilor să înceapă cu nivelul **A**. Cei care rezolvă fără dificultate problemele de nivelul **A**, vor continua cu problemele de nivelul **B**, iar cei care rezolvă cu uşurinţă şi aceste probleme vor continua rezolvînd probleme de nivelul **C**. Problemele de nivelul **C** vizează atingerea performanţelor maxime, de aceea sînt destinate celor mai buni şi mai insistenţi.

Toate programele incluse în culegere au fost testate şi depănate în mediul de programare Turbo Pascal 7.0.

Sper ca această carte să fie utilă şi să devină un ABC practic de însuşire a programării şi a Turbo-Pascal-ului.

Exprim mulţumiri recenzenţilor, colegilor de catedră, studenţilor, profesorilor şcolari şi nu în ultimul rînd Mihaelei Rusu, elevă la Liceul „Mircea Eliade", pentru observaţiile şi propunerile costructive, care au fost de un real folos în definitivarea acestei lucrări.

Sugestiile, recomandările şi opiniile cititorilor sînt aşteptate pe adresa editurii.

Autorul

1 Tipuri de date simple şi operaţii de bază asupra lor

Sugestii teoretice

Tipul datei defineşte domeniul de definiţie al datei şi mulţimea operatorilor care se pot aplica asupra valorilor datei.

Constantă – mărime a cărei valoare nu se poate modifica pe parcursul execuţiei algoritmului.

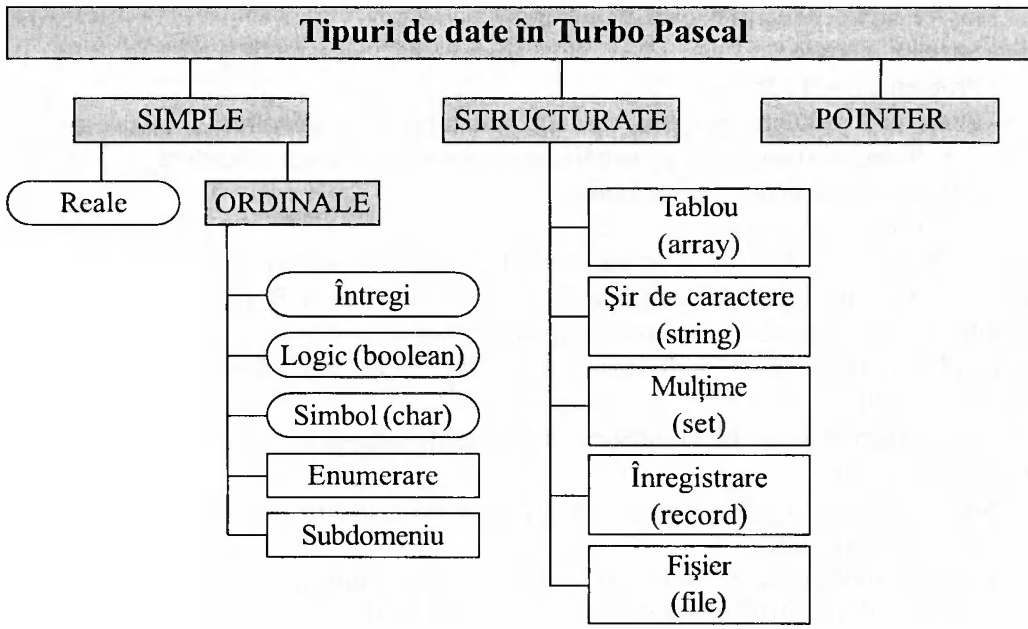
Variabilă – mărime a cărei valoare se poate modifica pe parcursul execuţiei algoritmului.

Identificator – secvenţă de caractere care începe cu o literă, ce poate fi urmată de una sau mai multe litere sau cifre, utilizată pentru a denumi constantele, variabilele, tipurile, procedurile şi funcţiile.

Operand – nume de date, constantă (de tip numeric sau şir de caractere), funcţie.

Expresie – combinaţie validă de operatori şi operanzi.

Comentariu – o consecutivitate arbitrară de caractere ce explică textul programului (În Turbo Pascal este textul cuprins între acolade, cu excepţia semnului $ sau între perechile de simbolului (* şi *)).



Notă: ( ) – tipuri predefinite (standard)

[ ] – tipuri definite de utilizator.

Observaţii

1. Tipurile simple sînt tipuri **scalare**, în sensul că asupra mulţimii valorilor fiecărui tip este definită relaţia de ordine. Deci, pentru orice două elemente e_1 şi e_2 ale mulţimii respective are loc o singură relaţie de forma $e_1 R e_2$, unde $R \in \{<, >, =\}$.
2. Pentru fiecare element al mulţimii unui tip ordinal există un unic predecesor (în afară de primul element) şi un unic succesor (în afară de ultimul element).

Tipuri de date simple standard

	Identificatorul tipului	Reprezentarea internă	Domeniul valorilor admisibile
Tipuri întregi	`byte`	pe 8 biţi (fără semn)	0 ... 255
	`word`	pe 16 biţi (fără semn)	0 ... 65535
	`shortint`	pe 8 biţi (cu semn)	–128 ... 127
	`integer`	pe 16 biţi (cu semn)	–32 768 ... 32 767
	`longint`	pe 32 biţi (cu semn)	–2147483648 ... 2147483647
Tipuri reale	`single`	pe 4 octeţi, $e_r = 10^{-7}$	$1{,}5 \cdot 10^{-45} - 3{,}4 \cdot 10^{38}$
	`real`	pe 6 octeţi, $e_r = 10^{-11}$	$2{,}9 \cdot 10^{-39} - 1{,}7 \cdot 10^{38}$
	`double`	pe 8 octeţi, $e_r = 10^{-15}$	$5 \cdot 10^{-324} - 1{,}7 \cdot 10^{308}$
	`extended`	pe 10 octeţi, $e_r = 10^{-19}$	$3{,}4 \cdot 10^{-4932} - 1{,}1 \cdot 10^{4932}$
Tipul logic	`boolean`	pe 1 octet	*true*, *false*
Tipul simbol	`char`	pe 1 octet	Toate simbolurile codului ASCII

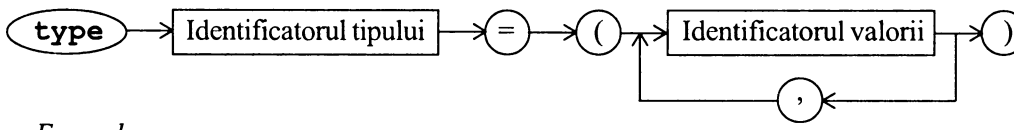
Observaţie

Fie v valoarea exactă a unei mărimi, $\bar{v}$ valoarea aproximativă a aceleiaşi mărimi.

Atunci: *eroarea absolută* este $e_a = |v - \bar{v}|$; *eroarea relativă* este $e_r = \frac{e_a}{|v|}$.

Tipuri de date simple definite de utilizator

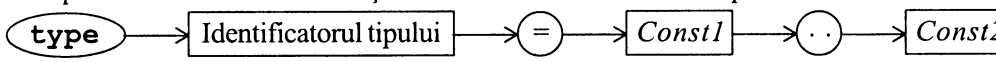
• Tipul **enumerare** se defineşte prin enumerarea identificatorilor ce reprezintă valorile mulţimii tipului:



Exemplu:

```
type saptamina=(luni, marti, miercuri, joi, vineri, simbata, duminica);
```

• Tipul **subdomeniu** se defineşte ca un subdomeniu al unui tip ordinal:



Const1 şi *Const2* (*Const1* < *Const2*) sînt constante ce reprezintă respectiv limita inferioară şi limită superioară a tipului respectiv.

Exemplu:

```
type zi=1..31; virsta=1..150; zi_lucr=luni..vineri;
```

Operatori, funcţii şi proceduri asupra tipurilor simple

	Identificatorul operatorului sau funcţiei	Tipul mărimilor sau parametrilor acceptaţi	Tipul rezultatului	Semnificaţia
Operatori aritmetici	+	întregi sau reali	real, dacă cel puţin o mărime este reală, altfel – întreg	returnează suma a două mărimi
	–	întregi sau reali	real, dacă cel puţin o mărime este reală, altfel – întreg	returnează diferenţa a două mărimi
	*	întregi sau reali	real, dacă cel puţin o mărime este reală, altfel – întreg	returnează produsul a două mărimi
	/	întregi sau reali	real	returnează rezultatul împărţirii a două mărimi
	DIV	întregi	întreg	`x` **DIV** `y` returnează cîtul obţinut la împărţirea lui *x* la *y*
	MOD	întregi	întreg	`x` **MOD** `y` returnează restul obţinut la împărţirea lui *x* la *y*
Funcţii şi proceduri matematice	Funcţia `ABS(x)`	întreg sau real	coincide cu tipul lui `x`	returnează valoarea absolută a lui *x*
	Funcţia `SQR(x)`	întreg sau real	coincide cu tipul lui `x`	returnează pătratul lui *x*
	Funcţia `SQRT(x)`	întreg pozitiv sau real pozitiv	real	returnează rădăcina pătrată a lui *x*
	Procedura `INC(x)`	întreg	întreg	valoarea lui *x* devine *x+1*
	Procedura `INC(x,y)`	întregi	întreg	valoarea lui *x* devine *x+y*
	Procedura `DEC(x)`	întreg	întreg	valoarea lui *x* devine *x-1*
	Procedura `DEC(x,y)`	întregi	întreg	valoarea lui *x* devine *x-y*

	Identificatorul operatorului sau funcției	Tipul mărimilor sau parametrilor acceptați	Tipul rezultatului	Semnificația
Funcții matematice	Funcția `TRUNC(x)`	întreg sau real	întreg	returnează partea întreagă a lui x, înlăturînd partea neîntreagă (fracționară)
	Funcția `ROUND(x)`	întreg sau real	întreg	returnează valoarea lui x rotunjită la cel mai apropiat întreg
	Funcția `INT(x)`	întreg sau real	real	returnează partea întreagă a lui x
	Funcția `FRAC(x)`	întreg sau real	real	returnează partea neîntreagă a lui x
	Funcția `SIN(x)`	întreg sau real	real	returnează valoarea $\sin x$
	Funcția `COS(x)`	întreg sau real	real	returnează valoarea $\cos x$
	Funcția `ARCTAN(x)`	întreg sau real	real	returnează $\operatorname{arctg} x$
	Funcția `EXP(x)`	întreg sau real	real	returnează valoarea e^x
	Funcția `LN(x)`	întreg sau real	real	returnează valoarea $\ln x$
Operatori de comparație	Operatorul =	2 parametri de același tip: numeric, simbol sau logic	logic	x = y returnează *true*, dacă $x = y$, altfel – *false*
	Operatorul <>	2 parametri de același tip: numeric, simbol sau logic	logic	x <> y returnează *true*, dacă $x \neq y$, altfel – *false*
	Operatorul >	2 parametri de același tip: numeric, simbol sau logic	logic	x > y returnează *true*, dacă $x > y$, altfel – *false*
	Operatorul <	2 parametri de același tip: numeric, simbol sau logic	logic	x < y returnează *true*, dacă $x < y$, altfel – *false*
	Operatorul <=	2 parametri de același tip: numeric, simbol sau logic	logic	x <= y returnează *true*, dacă $x \leq y$, altfel – *false*
	Operatorul >=	2 parametri de același tip: numeric, simbol sau logic	logic	x >= y returnează *true*, dacă $x \geq y$, altfel – *false*

	Identificatorul operatorului sau funcției	Tipul mărimilor sau parametrilor acceptați	Tipul rezultatului	Semnificația
Operatori logici	Operatorul **NOT**	logic	logic	**NOT** *x* returnează negația lui *x*
	Operatorul **AND**	logici	logic	x **AND** y returnează conjuncția mărimilor *x*, *y*
	Operatorul **OR**	logici	logic	x **OR** y returnează disjuncția mărimilor *x*, *y*
	Operatorul **XOR**	logici	logic	x **XOR** y returnează *false*, dacă *x* și *y* au aceeași valoare, *true*, dacă au valori diferite
Alte funcții	Operatorul **NOT**	întreg	logic	returnează *true*, dacă *x* este impar, *false*, dacă *x* este par
	Funcția `ORD(c)`	ordinal	întreg (byte)	returnează numărul de ordine al valorii *c*
	Funcția `CHR(i)`	întreg (byte)	simbol (caracter)	returnează caracterul cu numărul de ordine *i* în cadrul setului de caractere (ASCII)
	Funcția `PRED(c)`	ordinal	coincide cu tipul lui c	returnează valoarea ce precede valoarea lui *c*
	Funcția `SUCC(c)`	ordinal	coincide cu tipul lui c	returnează valoarea ce urmează după valoarea lui *c*
	Funcția `UPCASE(c)`	simbol	simbol	returnează litera mare corespunzătoare lui *c*. De exemplu, `UPCASE('d')` returnează `'D'` la fel ca și `UPCASE('D')`

Observații

1. Dacă c_1 și c_2 sînt două caractere, atunci $c_1 < c_2$ dacă și numai dacă $\mathtt{ORD}(c_1) < \mathtt{ORD}(c_2)$.
2. Prioritățile de aplicare a operatorilor:
   1. **NOT**.
   2. *, /, **AND**, **DIV**, **MOD**.
   3. +, –, **OR**, **XOR**.
   4. =, <, >, <>, >=, <=.
3. Numerele reale se pot reprezenta sub forma $m\mathrm{E}\pm e$, unde *m* este mantisa, *e* este exponentul. De exemplu, 3.1415E+15 semnifică $3{,}1415 \cdot 10^{15}$, iar 2.014E–20 semnifică $2{,}014 \cdot 10^{-20}$.

Structura generală a unui program Pascal

Antetul programului	`program ...`	
Secțiunea declarativă	`uses...`	Zona declarațiilor unităților de program
	`label...`	Zona declarațiilor etichetelor
	`const...`	Zona declarațiilor constantelor
	`type...`	Zona declarațiilor de tip
	`var...`	Zona declarațiilor de variabile
	`function...`	Zona declarațiilor de subprograme
	`procedure...`	

Secțiunea instrucțiunilor	`begin`
	`...`
	`end.`	

Instrucțiunea de atribuire. Proceduri de intrare-ieşire

Observație

Parantezele drepte indică parametrii opționali.

:= este instrucțiunea de atribuire şi se utitizează pentru modificarea valorii unei variabile.

De exemplu, instrucțiunile `a:=5; b:=a+2` atribuie variabilei `a` valoarea 5, iar variabilei `b` – valoarea 7 (deoarece `a` = 5).

READ ([f,], v_1 [, v_2, ..., v_n]) – citeşte datele din cîmpurile fişierului de intrare f şi le atribuie respectiv variabilelor v_1, v_2, ..., v_n.

Separatori de cîmp sînt considerate blancurile (spațiile). La sfîrşitul procedurii se avansează la cîmpul următor.

f reprezintă fişierul de intrare; dacă f lipseşte, atunci implicit se consideră fişierul standard de intrare *Input* (în acest caz citirea are loc de la tastatură).

READLN[([f,] v_1 [, v_2, ..., v_n])] – este similară procedurii `READ`, însă la sfîrşitul citirii se avansează la linia următoare.

WRITE ([f,] p_1 [, p_2, ..., p_n]) – scrie în fişierul f valorile parametrilor p_1, p_2, ..., p_n.

Dacă f lipseşte, atunci implicit se consideră fişierul standard de ieşire *Output* (în acest caz afişarea are loc la ecran).

WRITELN[([f,] p_1[, p_2, ..., p_n])] – este similară proceduri `WRITE`, însă la sfîrşitul scrierii se avansează la linia următoare.

Observații

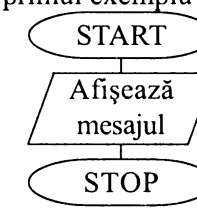
1. În urma execuției instrucțiunii `READLN` (sau `WRITELN`) se va avansa la linia următoare.
2. Fiecare dintre parametrii procedurilor `WRITE` și `WRITELN` poate avea una din următoarele forme de reprezentare:
   a) X
   b) $X:m$
   c) $X:m:f$,
   unde X este expresia (sau parametrul) a cărei valoare va fi scrisă, m reprezintă numărul minim de caractere ce vor fi scrise, iar f este folosit doar în cazul cînd valoarea lui X este de tip real și semnifică numărul cifrelor zecimale ale părții fracționare. Dacă $n(X)$ este numărul de cifre ale lui X și $n(X)<m$, atunci se vor scrie $m-n(X)$ spații de debut.
3. Valorile logice nu pot fi citite.
4. Valorile tipului enumerare nu pot fi citite sau afișate.

Probleme rezolvate

❶ Să se scrie un program care afișează la ecran mesajul „Acesta este primul exemplu".

Rezolvare:

```
program Exemplul1;
BEGIN
   write('Acesta este primul exemplu');
END.
```



❷ Să se scrie un program care calculează și afișează la ecran valoarea expresiei $23\cdot 5-4\sqrt{29}+2{,}71^2$.

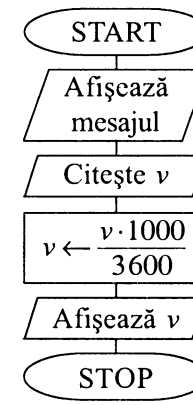
Rezolvare:

```
program Exemplul2;
BEGIN
   write(23*5-4*sqrt(29)+sqr(2.71));
END.
```

❸ Să se scrie un program care citește viteza v (în kilometri pe oră) de la tastatură și o afișează transformată în metri pe secundă.

Rezolvare:

```
program Exemplul3;
var v:real;
BEGIN
   write('Introdu viteza in km/h: ');
   readln(v);
   v:=v*1000/3600;
   write('Viteza in metri pe secunda: ',v);
END.
```



❹ Să se scrie un program care calculează valoarea expresiei $y = \arcsin x + 2\arccos x - \log_3 x + \sqrt[7]{5x}$ pentru x dat.

Rezolvare:

Observaţii

1. Luînd în considerație DVA al expresiei y, vom considera $x \in (0, 1]$.

2. Turbo Pascal, în afară de arctangentă, nu conţine funcţii standard pentru calcularea funcţiilor trigonometrice inverse, de aceea vom utiliza formulele:

$\arcsin x = \operatorname{arctg}\frac{x}{\sqrt{1-x^2}}$, $x \in (-1, 1)$ $(\arcsin(-1) = -\frac{\pi}{2}, \arcsin 1 = \frac{\pi}{2})$;

$\arccos x = \operatorname{arctg}\frac{\sqrt{1-x^2}}{x}$, $x \in (-1, 0) \cup (0, 1]$ $(\arccos(-1) = \pi, \arccos 0 = \frac{\pi}{2})$.

3. Din aceleaşi considerente vom utiliza formulele $a^b = e^{b \cdot \ln a}$, $\log_b a = \frac{\ln a}{\ln b}$.

4. $\sqrt[7]{5x} = (5x)^{\frac{1}{7}}$.

```
program Exemplul4;
var y,x:real;
BEGIN
   write('Introdu valoarea lui x: ');
   readln(x);
   y:=arctan(x/sqrt(1-x*x))+2*arctan(sqrt(1-x*x)/x)-ln(x)/ln(3)
      +exp(1/7*ln(5*x));
   write('f(',x,')=',y);
END.
```

❺ Să se scrie un program care calculează cîtul şi restul împărţirii la 7 a unui număr natural dat n.

Rezolvare:

```
program Exemplul5;
var n, cit, rest: integer;
BEGIN
   write('Introdu un numar natural: ');
   readln(n);
   cit:=n div 7;
   rest:=n mod 7;
   writeln('Restul impartirii lui  ', n, '  la 7 este ', rest);
   writeln('Citul impartirii lui  ', n, '  la 7 este ', cit);
END.
```

❻ Să se scrie un program care calculează suma cifrelor unui număr natural dat n de trei cifre.

Rezolvare:

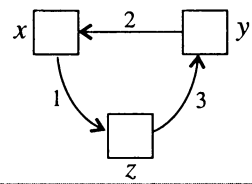
```
program Exemplul6;
var n, suma: integer;
BEGIN
   write('Introdu un numar natural de trei cifre (de la 100 la 999): ');
   readln(n);
```

```
    suma:= n div 100 + n div 10 mod 10 + n mod 10;
    write('Suma cifrelor numarului ', n,' este ', suma);
END.
```

❼ Să se scrie un program care schimbă între ele valorile a două variabile x şi y de tip real, citite de la tastatură, şi apoi afişează valorile lor. De exemplu, dacă $x = 2{,}3$ şi $y = -3{,}4$, în urma execuţiei vom avea $x = -3{,}4$ şi $y = 2{,}3$.

Observaţie

Vom utiliza schema:



Rezolvare:

```
Program Exemplul7;
uses Crt; {Declaram unitatea de program Crt}
var x, y, z: real;
BEGIN
   ClrScr; {Procedura ClrScr a unitatii Crt curata ecranul}
   write('Introdu x: '); readln(x)
   write('Introdu y: ' ); readln(y);
   z:=x;
   x:=y;
   y:=z;
   writeln('Dupa schimbare: ');
   write('x= ',x,' y= ',y);
END.
```

Observaţie

Există alt algoritm pentru schimbarea valorilor a două variabile numerice, fără utilizarea unei variabile auxiliare. Astfel, secvenţa de schimbare poate fi substituită cu secvenţa:

```
x:=x+y;
y:=x-y;
x:=x-y;
```

❽ Să se scrie un program care citeşte de la tastatură măsurile a două unghiuri (exprimate în grade, minute şi secunde), apoi afişează suma acestor măsuri. De exemplu, pentru măsurile 23°19′47″ şi 38°57′26″ obţinem suma 62°17′13″.

Rezolvare:

```
program Exemplul8;
uses Crt;
var gr1, gr2, min1, min2, min, sec1, sec2, sec: integer;
BEGIN
   ClrScr;
   write('Introdu masura unghiului 1: ');
   readln(gr1, min1, sec1);
   write('Introdu masura unghiului 2: ');
```

```
    readln(gr2, min2, sec2);
    sec:= sec1 + sec2; {se aduna secundele}
    min:= min1 + min2 + sec div 60; {se aduna minutele cu citul
      totalului secundelor la 60}
    sec:= sec mod 60; {secunde ramin atitea cit este restul impartirii
      totalului la 60}
    gr:= gr1 + gr2 + min div 60;
    min:= min mod 60;
    write('Suma: ',gr,' grade ',min,' minute ',sec,' secunde');
END.
```

Observaţie

Problema poate fi rezolvată şi prin altă metodă: iniţial se transformă ambele măsuri în secunde.

Exerciţii şi probleme propuse

A

1. Să se determine tipul şi valoarea expresiei scrise în limbajul Pascal pentru $a = 2$, $b = 5$, $c = 3$:

a) `a+b-c*a;`
b) `a+c/b-1;`
c) `b+c<2*a-c;`
d) `(a<b) and (c<a);`
e) `(a=b) or (b>c);`
f) `abs(a-c)>=1;`
g) `sqr(b+c)<>16;`
h) `not (a+b+c>10);`
i) `(a-b>c) xor (c<b-a);`
h) `a mod b<>0.`

2. Fie declaraţiile:

```
var a, b: real;
    m, n: integer;
    c: char;
    e: (doi, trei, patru, cinci);
    s: 'b'..'h';
```

Să se determine tipul expresiei scrise în limbajul Pascal:

a) `int(a+m);`
b) `sqrt(m);`
c) `round(a/(b+n));`
d) `ord(c);`
e) `succ(e);`
f) `ord(s);`
g) `trunc(a/b).`

3. Să se scrie cum se reprezintă în Pascal numărul:

a) 23,51;
b) 3!;
c) XXVI;
d) –9(3);
e) $\frac{3}{4}$;
f) $\sqrt{31}$;
g) $\sqrt[7]{211}$;
h) $-3{,}14 \cdot 10^{-4}$.

4. Să se scrie în Pascal expresia:

a) $ax^2 + bx + c$;
b) $\frac{3x^2 + 7}{2x - 1}$;
c) $\sqrt{2x^2 + xy}$;
d) $a\cos 2x - \sin y$;
e) $\operatorname{tg} x + \operatorname{ctg} y$;
f) $e^{2x-1} + \log_3 y$;
g) $\arcsin 5x - \arccos^3 2y$.

5. Să se scrie în limbaj matematic obişnuit:

a) `(A*X-B)/2;` b) `sqr(2*x+1)-exp(3);` c) `ln(5)/ln(2);`

d) `sqrt(1/3+2*x*sqr(y));` e) `exp(2/3*ln(y))`.

6. Să se calculeze valoarea expresiei (a, b, c, d, e, x, y se consideră date):

a) $ax^2 + bx + c$; b) $\ln 2 - \operatorname{arctg} 3$; c) $\arccos 0{,}23$;

d) $\arcsin(-0{,}78)$; e) $\operatorname{arctg}(-2)$.

7. Se dau două numere întregi. Să se determine suma şi diferenţa lor.

8. Se dau două numere întregi. Să se determine produsul şi cîtul lor.

9. Se dau două numere naturale. Să se determine media aritmetică şi media geometrică a acestora.

10. Se dă lungimea laturii unui pătrat. Să se afle perimetrul şi aria pătratului.

11. Se dă lungimea muchiei unui cub. Să se afle aria şi volumul cubului.

12. Se dă raza unui cerc. Să se afle lungimea cercului şi aria discului mărginit de acest cerc.

13. Se dă lungimea unui cerc. Să se afle raza cercului şi aria discului mărginit de acest cerc.

14. Se dă numărul natural n şi numărul real r.

a) Să se afle perimetrul şi aria poligonului regulat cu n laturi, dacă raza cercului circumscris lui este r.

b) Să se afle măsura unui unghi, numărul diagonalelor poligonului regulat cu n laturi şi raza cercului înscris în el, dacă raza cercului circumscris lui este r.

15. Se dau punctele $A(x1, y1)$, $B(x2, y2)$. Să se calculeze distanţa AB şi coordonatele mijlocului segmentului AB.

16. Se dă numărul natural n ($n < 10000$). Să se afişeze:

a) ultima cifră a acestui număr;

b) penultima cifră a acestui număr;

c) restul şi cîtul împărţirii acestui număr la 9.

17. Se dau numerele naturale n şi d (mai mici decît 10000), unde $n > d$.

Să se afişeze:

a) restul împărţirii lui n la d;

b) cîtul împărţirii lui n la d.

18. Ce va afişa la ecran instrucţiunea:

a) `writeln(Chr(Ord('a')+4));`

b) `writeln(Pred(Succ(10)))`?

19. Se dă numărul real n ($n < 10000$). Să se afişeze:
a) partea întreagă a lui n;
b) partea fracţionară a lui n;
c) rotunjirea lui n la întregi.

20. Se dau numerele reale x şi y. Să se afişeze TRUE dacă $x > y$, altfel – FALSE.

21. Se dă numărul n, care reprezintă măsura în grade a unui unghi orientat format de semidreapta [OM cu semiaxă pozitivă Ox a sistemului de axe ortogonale xOy. În ce cadran se află punctul M?

22. Se dă un număr natural m. Să se transforme în radiani măsura m în grade a unui unghi.

23. Se dă un număr real pozitiv r. Să se transforme în grade măsura r în radiani a unui unghi.

24. Se dă numărul natural n. Să se calculeze cîte ore, minute, secunde sînt în:
a) n zile (o zi = 24 ore);
b) n săptămîni;
c) luna mai.

25. Se dă numărul natural n.
a) Să se transforme n metri în centimetri.
b) Să se transforme n kilograme în miligrame.
c) Cîte tone întregi sînt în n kilograme?
d) Să se transforme n ani în luni, săptămîni, zile.

26. Se dau numerele naturale a, p, s.
Să se afle cu cît va creşte timp de a ani suma de s lei depusă la o bancă, dacă dobînda anuală este de p procente.

27. Se dau variabilele numerice a şi b. Să se schimbe între ele valorile acestor variabile.

28. Se dau variabilele numerice a, b şi c. Să se schimbe între ele valorile acestor variabile, astfel încît:
a să aibă valoarea lui b;
b să aibă valoarea lui c;
c să aibă valoarea lui a.

29. Să se afişeze următoarele date, respectînd formatul:

Struguri	100	kg
Mere	10	tone
Cartofi	250	kg
Varză	1 000	q

B

30. Se citesc de la tastatură 4 numere reale: a, b, c, d. Să se interschimbe circular de la stînga la dreapta valorile lor. De exemplu, pentru $a = 10$, $b = 20$, $c = 30$, $d = 40$, după interschimbare obţinem $a = 40$, $b = 10$, $c = 20$, $d = 30$.

31. Se dau numerele naturale n şi d. Să se calculeze:

a) $1 + 2 + ... + n$;

b) $4 + 4 + d + 4 + 2d + 4 + 3d + ... + 4 + (n-1)d$;

c) $5 + 5d + 5d^2 + 5d^3 + ... + 5d^{n-1}$.

32. Să se scrie un program care calculează diferenţa măsurilor a două unghiuri (exprimate în grade, minute şi secunde). Considerăm prima măsură mai mare decît a doua.

33. Se dă numărul n de tip integer. Să se calculeze suma cifrelor acestui număr.

34. Se dă numărul n de tip integer. Să se afişeze răsturnatul numărului n.

35. Se dau numerele naturale a şi b, unde $99 < a < 1000$, $b < 10$. Să se scrie un algoritm ce ar afişa scrierea numărului a în baza b.

36. Se dă un număr natural n mai mic decît 27. Să se afişeze litera a cărui număr de ordine în alfabetul latin este n.

37. Se dă o literă a alfabetului latin. Să se afişeze numărul de ordine al ei în acest alfabet. De exemplu, pentru litera M se va afişa 13.

38. O bancă dă o dobîndă anuală de p%.

a) Ce sumă de bani va avea după x ani un client care a depus S_0 lei? (p, x şi S_0 sînt date.)

b) Peste cîţi ani un client care a depus S_0 lei va avea pe cont S_f lei? (p, x şi S_f sînt date.)

39. Să se determine care este cel mai mare exponent al puterii numărului 16 ce poate fi calculată, memorizînd-o într-o variabilă de tip:

a) `integer`; b) `word`; c) `LongInt`.

C

40. Se dă o sumă de lei. Să se convertească suma într-un număr minim de bancnote.

Notă. În Republica Moldova există bancnote de 1000 lei, 500 lei, 200 lei, 100 lei, 50 lei, 20 lei, 10 lei, 5 lei, 1 leu.

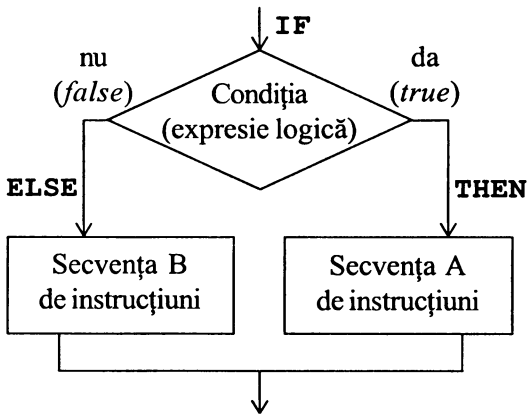
41. Se dau numerele întregi m şi n. Fără a compara, să se afişeze numărul mai mic.

42. Se dau numerele întregi m şi n. Fără a compara, să se afişeze numărul mai mare.

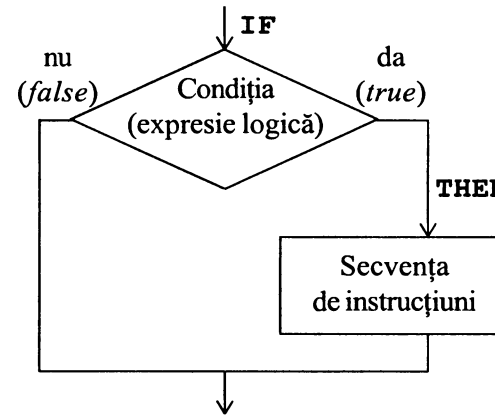
2 Structuri ramificate (alternative)

Sugestii teoretice

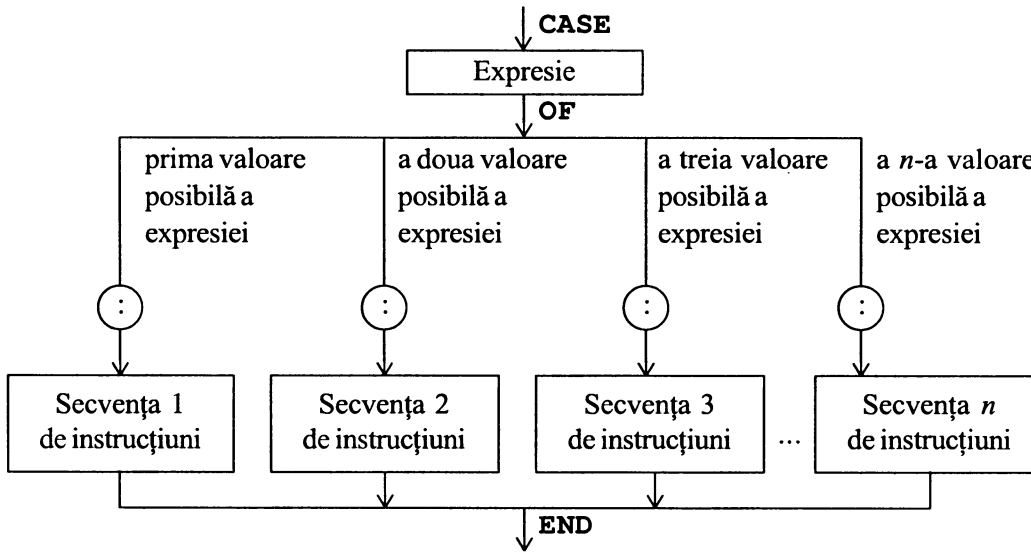
Decizia cu două alternative posibile



Decizia cu o alternativă posibilă



Decizia cu mai mult de două alternative posibile



Observaţii

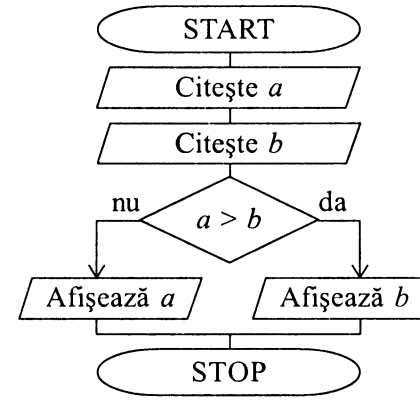
1. Expresia dintre CASE şi OF trebuie să fie de tip ordinal.
2. În cazul în care se doreşte ca una şi aceeaşi secvenţă de instrucţiuni să se execute pentru cîteva valori, acestea se vor delimita prin virgulă.

Probleme rezolvate

❶ Să se scrie un program care citeşte de la tastatură două numere întregi şi afişează la ecran numărul mai mic.

Rezolvare:

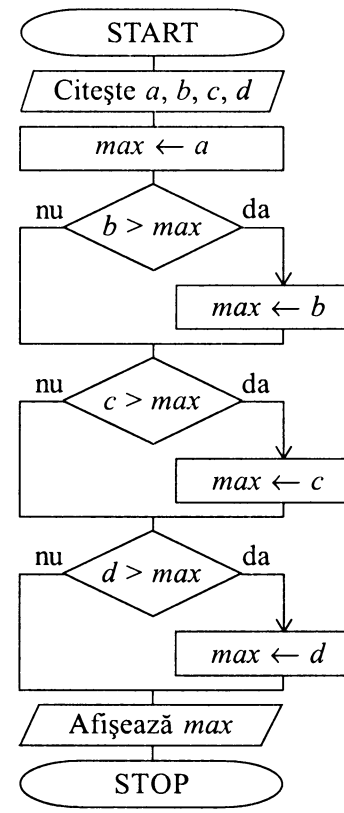
```
program Exemplul1;
uses Crt;
var a, b: integer;
BEGIN
   ClrScr;
   write('Introdu primul numar: ');
   readln(a);
   write('Introdu numarul al doilea: ');
   readln(b);
   write('Numarul mai mic este ');
   if a>b then write(b) else write(a);
   readkey; {Procedura unitatii Crt.
      Citeste de la tastatura un simbol}
END.
```



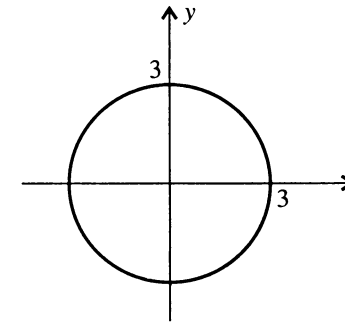
❷ Să se scrie un program care citeşte de la tastatură 4 numere întregi şi afişează la ecran numărul maximal.

Rezolvare:

```
program Exemplul2;
uses Crt;
var a, b, c, d, max: integer;
BEGIN
   ClrScr;
   write('Introdu 4 numere: ');
   readln(a, b, c, d);
   max:=a;
   if b>max then max:=b;
   if c>max then max:=c;
   if d>max then max:=d;
   write('Cel mai mare numar este ', max);
   readkey;
END.
```



❸ Să se scrie un program ce verifică dacă punctul $A(x, y)$, unde x, y sînt numere reale date, aparține domeniului colorat.




```
program Exemplul3;
uses Crt;
var x, y: real;
BEGIN
   ClrScr;
   write('Introdu coordonatele x şi y
     ale punctului A: ');
   readln(x, y);
   if (sqr(x)+sqr(y)<=9) and (x>0) and (y<0)
         then write('Apartine')
         else write('Nu apartine');
   readkey;
END.
```

Observații

1. Am utilizat faptul că $A(x, y)$ aparține cercului sau interiorului lui dacă $x^2 + y^2 \leq 3^2$.
2. Deoarece operatorii de comparație au ordinul de prioritate mai mic decît operatorul AND, am scris expresiile logice în paranteze.

❹ Să se scrie un program care citește de la tastatură numerele reale x și y și un operator *op* dintre +, –, *, /, apoi afișează valoarea expresiei x *op* y. De exemplu, pentru $x = 5$, $y = 11{,}2$, *op* = * programul va afișa 56.

Rezolvare:

```
program Exemplul4;
uses Crt;
var x, y, r: real;
    op: char;
BEGIN
   ClrScr;
   write('Introdu x:'); readln(x);
   write('Introdu y:'); readln(y);
   write('Introdu unul din operatorii +, -, *, /: '); readln(op);
   case op of
     '+': r:=x+y;
     '-': r:=x-y;
     '*': r:=x*y;
     '/': r:=x/y;
   end;
   write(x,op,y,'=',r);
   readkey;
END.
```

Observație

Programul nu va afișa nimic dacă se va introduce alt operator decît cei indicați. De aceea putem completa algoritmul inserînd înainte de **end** instrucțiunea **else** writeln('Operator nepermis ').

❺ Să se scrie un program care citeşte de la tastatură un număr întreg n, unde $|n|<10000$, apoi afişează numărul de cifre ale acestuia.

Rezolvare:

```
program Exemplul5;
uses Crt;
var n:integer;
BEGIN
   ClrScr;
   write('Introdu un numar intreg: ');
   readln(n);
   case n of
     0..9, -9..-1: writeln('Numarul este de o cifra');
     10..99, -99..-10: writeln('Numarul este de 2 cifre');
     100..999, -999..-100: writeln('Numarul este de 3 cifre');
     1000..9999, -9999..-1000: writeln('Numarul este de 4 cifre');
   end;
   readkey;
END.
```

❻ Se dă numărul natural n. Să se determine ultima cifră a numărului 7^n.

Rezolvare:

Analizînd puterile lui 7 ($7^1=7$, $7^2=49$, $7^3=343$, $7^4=2401$, $7^5=16807$...), observăm că ultima cifră poate fi 7, 9, 3, 1 şi este 7, dacă restul împărţirii exponentului puterii la 4 este 1; este 9, dacă restul împărţirii exponentului puterii la 4 este 2 etc.

```
program Exemplul6;
uses Crt;
var n:integer;
BEGIN
   ClrScr;
   write('Scrie exponentul puterii lui 7: ');
   readln(n);
   write('Ultima cifra a numarului 7^',n,' este ');
   case e mod 4 of
      1: write('7');
      2: write('9');
      3: write('3');
      0: write('1');
   end;
   readkey;
END.
```

Exerciții și probleme propuse

A

1. Fie `x = 1, y = 4, a = 0`. Care va fi valoarea lui `a` după executarea instrucțiunii?
   a) `if x>y then a:=x+y else a:=x-y;`
   b) `if x<=y then a:=sqr(x)+sqrt(y) else a:=sqrt(x)+sqr(y);`
   c) `if x+y<x*y then a:=x+y else a:=x-y;`
   d) `if not (x+1>y) then a:=x;`
   e) `if a in[1..10] then a:=1 else a:=2.`

2. Fie `x = 9, y = 4, a = 2`. Care va fi valoarea lui `a` după executarea instrucțiunii?
   a) `if (sqr(x)>a) and (sqr(y)=4) then a:=1;`
   b) `if (sqrt(x)>a) or (sqr(y)=4) then a:=1;`
   c) `if true then a:=0;`
   d) `if not true then a:=1 else a:=0;`
   e) `if 5=5/5+4 then a:=a.`

3. Să se scrie în Pascal expresia logică:
   a) *a* este mai mare decît 3;
   b) *b* este diferit de 9;
   c) *a* este mai mic decît 3 şi mai mare decît 1;
   d) *b* este mai mic sau egal cu –5;
   e) *a* + *b* nu este număr par;
   f) *a* este egal cu 0 sau *b* este egal cu 3;
   g) cel puțin unul din numerele *a* şi *b* este negativ;
   h) *a* este cel mult egal cu pătratul lui *b*.

4. Fie programul:

```
program ramificator;
var n1, n2, n3, n_m: byte;
BEGIN
   n1:=8; n2:=9;
   readln(n3); n_m:=(n1+n2+n3) div 3;
   case n_m of
     1..4: writeln('Rau');
     5..6: writeln('Suficient');
     7..8: writeln('Bine');
     9..10: wtiteln('Excelent');
   end;
END.
```

   Ce valoare trebuie să i se atribuie lui `n3` pentru ca programul să afişeze:
   a) Suficient; b) Bine; c) Rau; d) Excelent?

5. Se dau numerele reale a şi b. Să se calculeze:

a) $\max(a, b)$; b) $\min(a, b)$; c) $\max(a, b) + \min(a, b)$;

d) $\max(a - b, b)$; e) $\min(a + b, a)$; f) $\max(2a, 3b)$.

6. Se dau numerele reale a, b şi c. Să se calculeze:

a) $\max(a, b, c)$; b) $\min(a, b, c)$;

c) $\dfrac{\max(a, b, c)}{\min(a, b, c)}$; d) $3 \cdot \max(a, b, c)$;

e) $4 - 2 \cdot \max(a, b, a + b - c)$; f) $[\max(a-b, b-c, a-c)]^2 - 1$.

7. Se dau numerele reale a şi b. Să se afişeze:

a) a, dacă $a > b$, în caz contrar, să se afişeze b;

b) b, dacă $b - a > a - b$, în caz contrar, să se afişeze $a - b$;

c) $a + b$, dacă $a = b$, în caz contrar, să se afişeze $b - a$;

d) a şi b, dacă a este diferit de b, în caz contrar, să se afişeze doar a.

8. Să se calculeze valoarea funcţiei:

a) $f(x) = \begin{cases} x^2, & x < -5, \\ x+1, & -5 \le x < 2, \\ x^3, & x \ge 2, \end{cases}$ b) $f(x) = \begin{cases} \cos x, & x < 0, \\ 4, & x = 0, \\ \sin x, & x > 0; \end{cases}$

c) $f(x) = \begin{cases} 3x + |x-1|, & x < 1, \\ 2, & 1 \le x < 6, \\ \log_3 x, & x \ge 6; \end{cases}$ d) $f(x) = \begin{cases} 1, & x \le 0, \\ x^2 + x, & 0 < x \le 2, \\ \sin \pi x, & x > 2. \end{cases}$

9. Se dau numerele întregi m şi n. Să se verifice dacă ele sînt consecutive.

10. Se dau numerele naturale diferite a, b, c.

a) Să se afişere numărul care nu este nici cel mai mare, nici cel mai mic.

b) Să se verifice dacă ele pot fi termeni consecutivi ai unei progresii aritmetice.

11. Să se scrie o expresie logică a cărei valoare va fi *true*, dacă propoziţia este adevărată, şi *false*, dacă propoziţia este falsă.

a) Numărul natural n de 2 cifre are cifra unităţilor identică cu cifra zecilor.

b) Numărul s este soluţie a ecuaţiei $ax + b = 0$ sau a ecuaţiei $cx^2 + dx + e = 0$.

c) Numărul întreg n este divizibil cu numărul m, dar nu este divizibil cu numărul k.

d) Numele a, b, c sînt pitagoriene, adică pot fi lungimile laturilor unui triunghi dreptunghic.

e) Nu există un triunghi cu laturile de lungimi a, b, c.

f) Ecuaţia $ax^2 + bx + c$ are exact o rădăcină pozitivă.

h) Numărul întreg n nu este de 3 cifre.

12. Se dau numerele naturale a, b, c.

Să se verifice dacă a, b, c sînt numere pitagoriene.

13. Se dau numerele naturale a şi b ($b < a$). Să se verifice dacă b divide a.

14. Să se rezolve ecuaţia (numerele a, b, c sînt date):

a) $ax + b = 0$; b) $ax^2 + bx + c = 0$.

15. Să se rezolve inecuaţia (numerele a, b, c sînt date):

a) $ax + b < 0$; b) $ax + b \geq 0$; c) $ax^2 + bx + c \leq 0$; d) $ax^2 + bx + c > 0$.

16. Să se determine numărul cadranului în care este situat punctul $P(x, y)$ (numerele întregi x şi y sînt date).

17. Să se reprezinte pe caiet într-un sistem de axe ortogonale xOy domeniul în care expresia logică returnează valoarea *true*:

a) `x-y+4<0;` b) `sqr(x)+sqr(y)-2*y-9>0;`

c) `2*x+y-2>=0;` d) `(x<0) and (x-y<1).`

18. Se dă numărul natural n. Să se verifice dacă n este:

a) par; b) divizibil cu 2 şi cu 3; c) divizibil cu 3 sau cu 4;

d) divizibil cu 3, dar nu este divizibil cu 4; e) nu este divizibil nici cu 3, nici cu 4.

19. Să se formulcze enunţul problemei care se rezolvă cu ajutorul algoritmului:

a)
```
program R1;
var x1,x2,y1,y2,d1,d2:real;
BEGIN
  write('Introdu x1,y1,x2,y2: ');
  readln(x1,y1,x2,y2);
  d1:=sqrt(x1*x1+y1*y1);
  d2:=sqrt(sqr(x2)+sqr(y2));
  if d1=d2 then write('Egal departate')
     else write('Nu sint egal departate');
END.
```

b)
```
program R2;
var a,b,d:integer;
BEGIN
  wrire('Introdu a, b, d: ');
  readln(a,b,d);
  if (a mod d=0) and (b mod d=0) then write('Este')
     else write ('Nu este');
END.
```

c)
```
program R3;
var x,y,z,t:integer;
BEGIN
  write('Introdu x, y, z: ');
  readln(x,y,z);
  if x<y then begin
     t:=x; x:=y; y:=t;
  end;
```

```
  if x<z then begin
     t:=x; x:=z; z:=t;
  end;
  if y<z then begin
     t:=y; y:=z; z:=t;
  end;
  write(x,' ',y,' ',z);
END.
```

d)
```
program R4;
var a,m,n:integer;
BEGIN
  write('Introdu un numar intreg de 2 cifre: ');
  readln(a);
  m:=a div 10; n:=a mod 10;
  if m>n then a:=a-n;
  if n>m then a:=a-m;
  write(a);
END.
```

e)
```
program R5;
var a,b,m:integer;
BEGIN
  write('Introdu 2 numere intregi: '); readln(a, b);
  if a>b then m:=a-b else m:=b-a;
  write('L=',m);
END.
```

20. Se dau numerele întregi a, b, c. Să se afişeze numerele în ordine:
a) crescătoare; b) descrescătoare.

21. Se dau numerele naturale a, b, c. Să se verifice dacă există un triunghi cu unghiurile de măsurile (în grade) a, b, c. În caz afirmativ, să se determine tipul triunghiului: echilateral, isoscel, scalen.

22. Se dă numărul natural n, $0 < n < 8$. Să se afişeze denumirea zilei corespunzătoare cifrei respective. De exemplu, pentru $n = 3$ se va afişa 'Miercuri'.

23. Se dă o literă. Să se verifice dacă ea este consoană sau vocală.

24. Se dă numărul natural n. Să se determine ultima cifră a produsului $1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$.

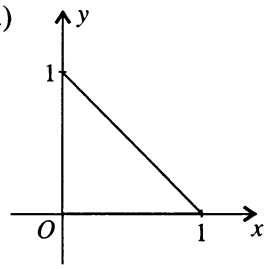
25. Se dau numerele naturale nenule a, b, c. Să se verifice dacă fracţia $\frac{a}{b}$ poate fi simplificată prin c. În caz afirmativ, să se afişeze fracţia simplificată.

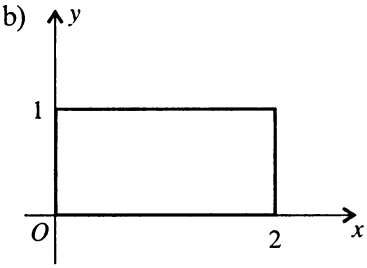
26. Să se reprezinte într-un sistem de axe ortogonale *xOy* domeniul în care expresia logică returnează valoarea *true*:
a) `(y-x+2>=0) and (y+x+2>=0) and (sqr(x)+sqr(y)<=4);`
b) `(y<=-x+1) and (y<x+1) and (y>=-x-1) and (y>=x-1);`

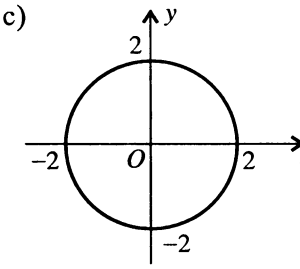
c) `(y+x-1>=0) and (y-x-1>=0) and (y+x+1<=0) and (y-x+1<=0) and (sqr(x)+sqr(y)<=1);`

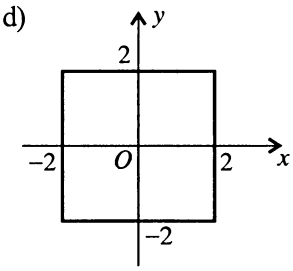
d) `(y>=1) or (y<=1) and (y>=0) and (y>=x) and (y>=-x).`

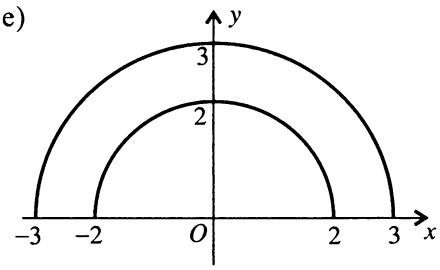
27. Să se verifice dacă punctul $P(x, y)$ (numerele întregi x şi y sînt date) aparţine domeniului colorat:

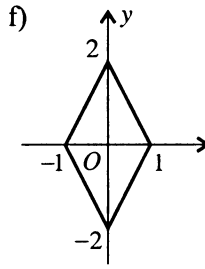
a) 

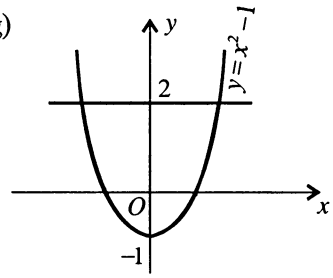
b) 

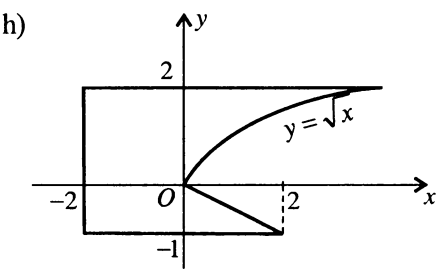
c) 

d) 

e) 

f) 

g) 

h) 

28. Bursa (B) studenţilor se calculează în funcţie de nota medie (n_m):

dacă $n_m < 5$, atunci $B = 0$;

dacă $5 \le n_m < 7$, atunci $B = 100$ (lei);

dacă $7 \le n_m < 8$, atunci $B = 20 \cdot n_m$ (lei);

dacă $8 \le n_m < 10$, atunci $B = 25 \cdot n_m$ (lei);

dacă $n_m = 10$, atunci $B = 300$ (lei).

Să se scrie un program care citeşte de la tastatură nota medie şi afişează la ecran bursa calculată în funcţie de această notă.

29. Fie $O_1(x_1, y_1)$ şi $O_2(x_2, y_2)$ centrele cercurilor $\mathcal{C}(O_1, R_1)$ şi $\mathcal{C}(O_2, R_2)$. Să se determine poziţiile relative ale cercurilor, fiind date numerele reale x_1, y_1, x_2, y_2, R_1, R_2.

30. Se dă numărul natural n, $n \in \{28, 29, 30, 31\}$. Să se afişeze denumirile lunilor cu numărul de zile n. De exemplu, pentru $n = 30$, se va afişa:
aprilie, iunie, septembrie, noiembrie.

31. Se dau numerele reale diferite a, b, c. Să se determine care dintre ele este situat pe axa numerică între celelalte două.

32. Se dau numerele a, b şi coordonatele carteziene ale unui punct. Să se determine poziţia punctului faţă de dreapta $y = ax + b$.

33. Se dau razele şi coordonatele carteziene ale centrelor a două cercuri. Să se determine poziţia relativă a cercurilor (interioare, exterioare, tangente interior, tangente exterior, secante, concentrice).

34. Se dau 4 numere. Să se determine dacă ele pot reprezenta lungimile laturilor unui paralelogram.

35. Conform calendarului japonez, fiecare an poartă numele unui animal. Fiecare denumire se repetă exact o dată la 12 ani. Deci, un ciclu este format din 12 ani cu următoarele denumiri de animale în această ordine: şobolan, bou, tigru, iepure, dragon, şarpe, cal, oaie, maimuţă, cocoş, cîine, porc. Ştiind că anul 2000 a fost anul Dragonului, să se scrie un algoritm care va citi de la tastatură anul (număr de patru cifre) şi va afişa denumirea lui conform calendarului japonez.

B

36. Se dau numerele naturale a, b, c. Să se verifice dacă există un triunghi ale cărui unghiuri au măsurile (în grade) egale cu a, b, c. În caz afirmativ, să se determine tipul triunghiului: ascuţitunghic, obtuzunghic, dreptunghic.

37. Se dau numerele reale pozitive a, b, c. Să se verifice dacă există un triunghi ale cărui laturi au lungimile (în aceeaşi unitate de măsură) egale cu a, b, c. În caz afirmativ, să se determine tipul triunghiului: echilateral, isoscel, scalen.

38. Se dau numerele reale pozitive a, b, c. Să se verifice dacă există un triunghi ale cărui laturi au lungimile (în aceeaşi unitate de măsură) egale cu a, b, c. În caz afirmativ, să se determine tipul triunghiului: ascuţitunghic, obtuzunghic, dreptunghic.
Indicaţie. Se va utiliza formula $a^2 = b^2 + c^2 - 2bc\cos(\widehat{b,c})$.

39. Se dau numerele a, b, c, d. Considerînd un sistem de axe ortogonale xOy, să se determine poziţia dreptelor $y = ax + b$, $y = cx + d$. În caz de intersecţie, să se determine coordonatele punctului de intersecţie.

40. Se dau numerele întregi m, n, p. Să se verifice dacă ele sînt consecutive.

41. Se dau numerele întregi m, n, p, q. Să se verifice dacă ele sînt consecutive.

42. Dintre numerele întregi date a, b, c, d, e să se selecteze:
a) maximum2 – numărul mai mic decît numărul maximal şi mai mare decît celelalte numere;
b) minimum2 – numărul mai mare decît numărul minimal şi mai mic decît celelalte numere.

43. Se dau numerele întregi a, b, c, d. Să se afişeze numerele în ordine:
a) crescătoare; b) descrescătoare.

44. Se dau numerele întregi z_n, l_n, a_n, z_c, l_c, a_c care reprezintă respectiv ziua, luna, anul naşterii unei persoane şi ziua curentă, luna curentă, anul curent. Să se determină vîrsta persoanei:
a) în zile; b) în luni împlinite; c) în ani împliniţi.

45. Se dau numerele naturale m şi n ($m \le 12$, $n < 60$), ce indică momentul de timp „ora m şi n minute". Să se determine peste cîte minute:
a) acelc orarului şi minutarului vor coincide;
b) acele orarului şi minutarului vor fi perpendiculare.

46. Se dau numerele pozitive a, b. Care dintre punctele coliniare M, N, P nu poate fi situat între celelalte două, dacă:
a) $MN = a$, $MP = b$; b) $MP = a$, $NP = b$?

47. Se dau numerele naturale a şi b, unde $b < 11$. Să se verifice dacă există un număr a cărui scriere în baza b este a. De exemplu, pentru $a = 543$ şi $b = 7$ răspunsul este pozitiv (deoarece $543 = 1404_7$), iar pentru $a = 543$ şi $b = 5$ răspunsul este negativ.

48. Se dă numărul natural n. Să se determine ultima cifră a numărului 2^n.

49. Se dau 3 litere mici ale alfabetului latin. Să se verifice dacă ele sînt consecutive.

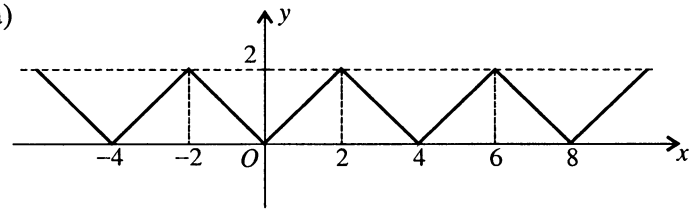
50. Se dau 4 litere mari ale alfabetului latin. Să se verifice dacă ele sînt consecutive.

51. Se dau numerele naturale m şi n. Să se determine:
a) cel mai mic număr de două cifre, format din cifrele unităţilor ale celor două numere (de exemplu, pentru $m = 112$ şi $n = 39$ se va afişa 29);
b) cel mai mare număr de două cifre, format din cifrele zecilor ale celor două numere; dacă numărul este de o cifră, atunci cifra zecilor se va considera 0 (de exemplu, pentru $m = 30$ şi $n = 9$ se va afişa 30, iar pentru $m = 24$ şi $n = 473$ se va afişa 72).

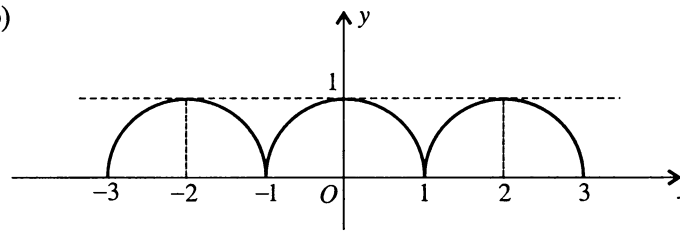
52. Se dă numărul natural n. Să se determine ultima cifră a sumei $2^n + 3^n + ... + 9^n$.

53. Se dă numărul real x. Să se calculeze valoarea funcției f în punctul x, dacă f este periodică şi are graficul reprezentat:

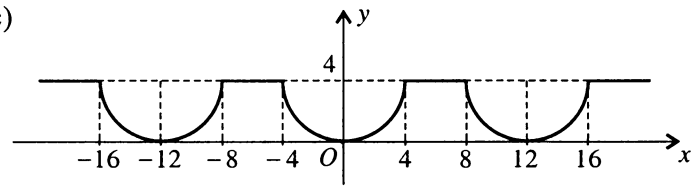
a)



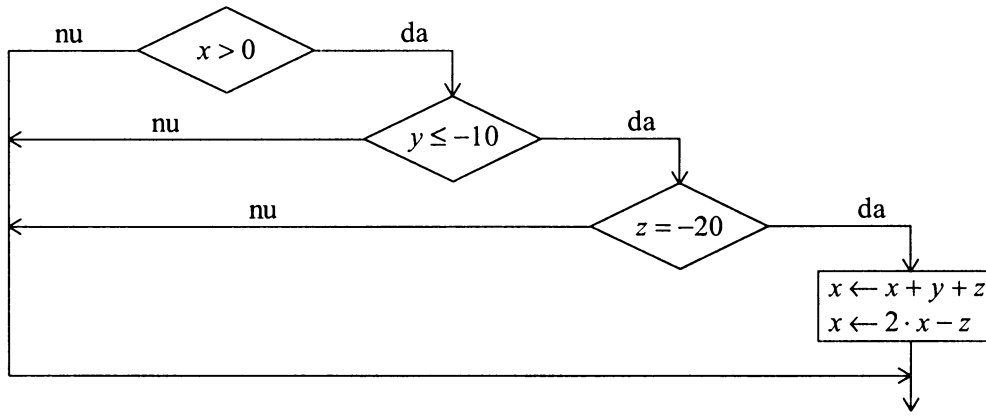
b)



c)



54. Să se scrie secvența în Pascal corespunzătoare următoarei scheme logice, utilizînd o singură dată instrucțiunea IF şi o singură dată instrucțiunea de atribuire.



55. Se dă un număr natural $n > 7$. Să se rezolve în $\mathbb{N}$ ecuația $5x + 3y = n$. De exemplu, pentru $n = 8$ soluția va fi $(1, 1)$.

56. Orice cîmp al tablei de şah se determină de 2 numere: primul – numărul verticalei (de la stînga la dreapta), al doilea – numărul orizontalei (de jos în sus).
Se dau numerele naturale a, b, c, d mai mici decît 9. Să se verifice dacă:
a) cîmpurile (a, b) şi (c, d) se află în aceeaşi linie (orizontală) sau în aceeaşi coloană (verticală);
b) cîmpurile (a, b) şi (c, d) sînt de aceeaşi culoare;
c) nebunul situat pe cîmpul (a, b) poate (dintr-o mişcare) ajunge pe cîmpul (c, d);
d) calul situat pe cîmpul (a, b) poate (dintr-o mişcare) ajunge pe cîmpul (c, d).

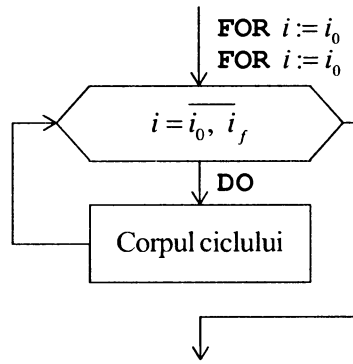
57. Se dau coordonatele carteziene ale unui punct şi ale vîrfurilor unui triunghi. Să se determine poziția punctului față de triunghi (aparține triunghiului, interiorului sau exteriorului).

58. Se dau coordonatele cartiziene ale 3 vîrfuri ale unui dreptunghi. Să se afle coordonatele celui de-al patrulea vîrf. De exemplu, pentru coordonatele date:
5, 1
1, 1
1, 7
se va afişa 5, 7.

3 Structuri repetitive

Sugestii teoretice

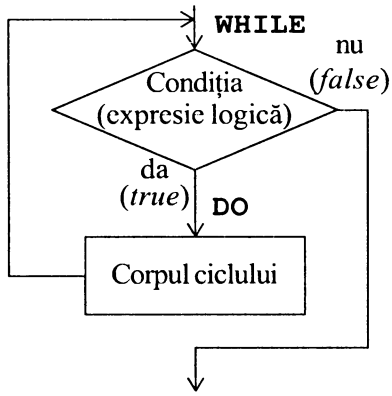
Ciclu cu parametru



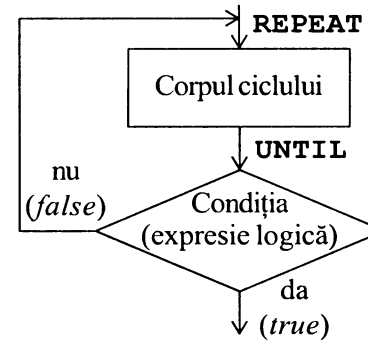
Observație

Variabila *i* este de tip ordinal.

Ciclu cu condiție anterioară



Ciclu cu condiție posterioară



Observație

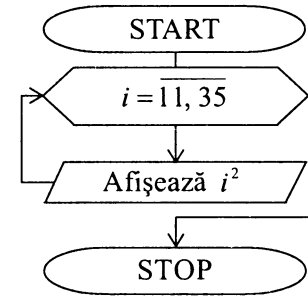
În cazul ciclului cu parametru şi al celui cu condiție anterioară, dacă corpul ciclului constă din mai mult de o instrucțiune, atunci acesta se încadrează între cuvintele-cheie **begin** şi **end**.

Probleme rezolvate

❶ Să se scrie un program ce calculează şi afişează la ecran pătratele numerelor 11, 12, 13, ..., 35.

Rezolvare:

```
program Exemplul1;
uses Crt;
var i: integer;
BEGIN
   ClrScr;
   for i:=11 to 35 do
   writeln(sqr(i));
   readkey;
END.
```



❷ Să se scrie un program care calculează şi afişează la ecran în ordine descrescătoare divizorii proprii ai unui număr natural dat n. De exemplu, pentru $n = 24$ obţinem divizorii 12, 8, 6, 4, 3, 2.

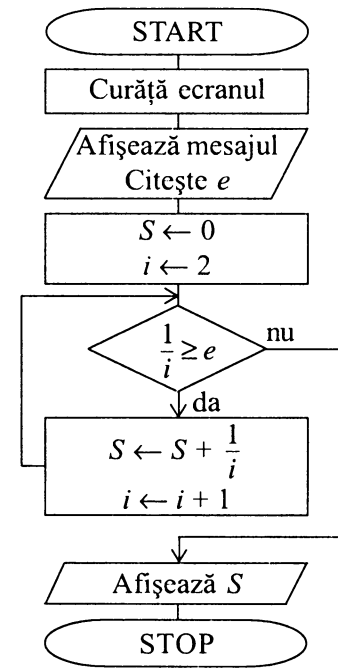
Rezolvare:

```
program Exemplul2;
uses Crt;
var n, i: integer;
BEGIN
   ClrScr;
   write('Introdu n: '); readln(n);
   for i:=n div 2 downto 2 do
     if n mod i=0 then writeln(i);
   readkey;
END.
```

❸ Fie suma $S=\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\ldots$ Să se scrie un program care calculează suma S cu exactitatea dată e (se va suma atît timp cît noul termen va fi mai mare sau egal cu e).

Rezolvare:

```
program Exemplul3;
uses Crt;
var i: integer; S: real;
BEGIN
   ClrScr;
   write('Introdu exactitatea: '); readln(e);
   S:=0; i:=2;
   while 1/i>=e do begin
   S:=S+1/i; i:=i+1;
   end;
   write('S= ',S);
   readkey;
END.
```



❹ Să se scrie un program care calculează factorialul unui număr natural dat n, $n < 10$. Amintim, că $n! = 1 \cdot 2 \cdot 3 \cdot ... \cdot n$.

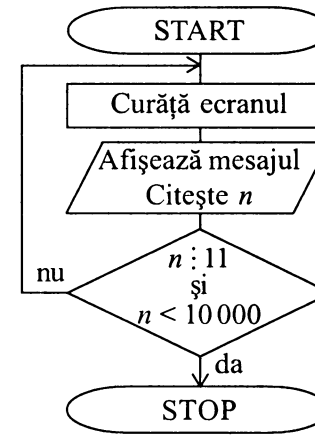
Rezolvare:

```
program Exemplul4;
uses Crt;
var i,n:integer;
    f:longint;
BEGIN
   ClrScr;
   write('Introdu n: '); readln(n);
   f:=1;
   for i:=1 to n do
     f:=f*i;
   writeln(n,'!=', f);
   readkey;
END.
```

❺ Să se scrie un program care va cere utilizatorului să introducă un număr întreg divizibil cu 11 şi mai mic decît 10000. Programul va rula pînă cînd utilizatorul va introduce un număr cu proprietatea menţionată.

Rezolvare:

```
program Exemplul5;
uses Crt;
var n: integer;
BEGIN
   repeat
     ClrScr;
     write('Introdu un numar divizibil cu 11
      mai mic decit 10000: ');
     readln(n);
   until (n mod 11=0) and (n<10000);
   writeln('Corect. Apasa orice tasta');
   readkey;
END.
```



❻ Se dă numărul natural n. Să se scrie descompunerea în factori primi a numărului n.

Rezolvare:

Vom calcula (numărul p) de cîte ori numărul i (iniţial $i = 2$) se conţine ca factor în n (de fiecare dată numărul n devine de i ori mai mic), după care vom mări valoarea lui i cu 1. Procesul va dura atît timp cît $n > 1$.

```
program Exemplul6;
uses Crt;
var n,i,p:word;
BEGIN
   ClrScr;
   write('Introdu un numar natural: ');
   readln(n);
   i:=2;
   while n>1 do begin
      if n mod i=0 then begin {i se contine ca factor}
```

```
        p:=0;
        repeat {calculam puterea lui i}
           inc(p);
           n:=n div i;
        until n mod i<>0;
        write(i,'^',p,'+');
     end;
     inc(i);
  end; {while}
  writeln(#8,' '); {scrie un spatiu in loc de ultimul +}
  readkey;
END.
```

❼ Se dă un număr natural format din cel mult 9 cifre. Să se determine de cîte ori se repetă în scrierea lui cifra unităţilor şi de cîte ori cifra zecilor.

```
program Exemplul7;
uses Crt;
var n:longint;
    u,z,nu,nz:byte; {u este cifra unitatilor, iar z - cifra zecilor}
BEGIN
  ClrScr;
  write('Introdu un numar natural (cu cel mult 9 cifre): ');
  readln(n);
  nu:=0; nz:=0; {nz este numarul de aparitii a cifrei zecilor}
  u:=n mod 10;
  z:=n div 10 mod 10;
  while n<>0 do begin
     if n mod 10=u then inc(nu);
     if n mod 10=z then inc(nz);
     n:=n div 10;
  end;
  writeln('Cifra unitatilor se repeta de ',nu,' ori');
  writeln('Cifra zecilor se repeta de ',nz,' ori');
  readkey;
END.
```

❽ Se dă numărul natural *n*. Să se afişeze la ecran a *n*-a cifră a numărului 14916253649... (sînt scrise consecutiv pătratele tuturor numerelor naturale pozitive).

Rezolvare:

```
program Exemplul8;
uses Crt;
var i,j,m,n,cif:integer;
    p,t:longint;{p - patratul numarului j}
BEGIN
  ClrScr;
  write('Introdu numarul n: ');
  readln(n);
  i:=0; j:=0;
  repeat
     inc(j);
     p:=sqr(j);
     t:=p;
```

```
      while t>0 do begin {numarul de cifre ale lui p le adaugam la i}
         t:=t div 10;
         inc(i);
      end;
   until i>=n;
   while i>=n do begin {daca i a intrecut n, atunci parcurgem inapoi cite
      o cifra din p, micsorindu-l pe i}
      cif:=p mod 10;
      p:=p div 10;
      dec(i);
   end;
   write('A ', n, '-a cifra a numarului 149162536496481100121...
      este ',cif);
   readkey;
END.
```

❾ *Biliard*

Să se simuleze mişcarea unei bile de biliard (ecranul va avea rolul mesei de biliard).

Rezolvare:

Litera „o" va reprezenta bila şi iniţial va fi situată în punctul de coordonate (x, y). Evident, cînd bila va atinge una dintre cele 2 margini orizontale (respectiv verticale), creşterea *dx* a orizontalei (respectiv *dy* a verticalei) îşi va scimba semnul.

```
program Exemplul9;
uses Crt;
var x,y,dx,dy:integer;
BEGIN
   ClrScr;
   dx:=1; dy:=1;
   writeln('Scrie coordonatele initiale ale bilei 1<X<80, 1<Y<25: ');
   readln(x,y);
   ClrScr;
   while not KeyPressed do begin
      gotoxy(x,y);
      write('o');
      delay(200);
      if (x=1) or (x=80) then dx:=-dx;
      if (y=1) or (y=25) then dy:=-dy;
      gotoxy(x,y);
      write(' ');
      x:=x+dx;
      y:=y+dy;
   end;
END.
```

Observaţie

Procedura `GotoXY(X,Y:byte)` poziţionează cursorul de text în coloana `X` şi linia `Y` a ecranului. Funcţia `KeyPressed` returnează valoarea *true* dacă a fost apăsată o tastă.

Probleme propuse

A

1. Să se tabeleze funcția $f(x)$ pe intervalul $[a, b]$ cu pasul h, dacă:

a) $f(x) = \ln(x+1)$, $a = 0$, $b = 2$, $h = 0{,}2$;

b) $f(x) = \arcsin(x)$, $a = -0{,}9$, $b = -0{,}1$, $h = 0{,}01$;

c) $f(x) = \log_4(x+1)^2$, $a = -3$, $b = 0$, $h = 0{,}3$;

d) $f(x) = \cos(x)$, $a = 0$, $b = 4$, $h = 0{,}5$;

e) $f(x) = \arccos(x) + \arcsin(x)$, $a = 0{,}1$, $b = 0{,}9$, $h = 0{,}1$;

f) $f(x) = \operatorname{arcctg}\left(-\frac{1}{\sqrt{x}}\right)$, $a = 0{,}1$, $b = 5$, $h = 0{,}2$;

g) $f(x) = e^{x^2}$, $a = -2$, $b = 2$, $h = 0{,}2$;

h) $f(x) = \sin(x)$, $a = -2$, $b = 4$, $h = 0{,}5$;

i) $f(x) = \sqrt{4 - x^2}$, $a = -2$, $b = 2$, $h = 0{,}2$;

j) $f(x) = \log_x 10$, $a = 2$, $b = 16$, $h = 0{,}5$;

k) $f(x) = x^2$, $a = -2$, $b = 2$, $h = 0{,}5$.

2. Se dau numerele naturale k şi a_1. Să se afişeze primii k termeni ai şirului definit prin formula recurentă $a_{n+1} = a_n + 4n$, unde a_n este termenul al n-lea.

3. Se dau numerele naturale k, a şi d. Să se calculeze produsul primilor k termeni ai progresiei aritmetice, dacă primul termen este a, iar raţia este d.

4. Să se calculeze $(2n)!$ pentru n dat.

5. Să se afişeze primii k termeni ai şirului definit de formula $a_n = n^2 + 1$, unde k este număr natural dat.

6. Se dau numerele naturale m şi n, unde $m < n$. Să se calculeze produsul numerelor mai mici decît n, divizibile cu m.

7. Să se determine dacă există patru numere naturale consecutive, astfel încît suma pătratelor lor este egală cu suma pătratelor următoarelor trei numere.

8. Să se calculeze cel mai mare divizor comun şi cel mai mic multiplu comun a:
a) două numere date; b) trei numere date; c) patru numere date.

9. Să se calculeze valoarea expresiei $\cos(x) + \cos\cos(x) + \ldots + \underbrace{\cos\cos\ldots\cos}_{\text{de } n \text{ ori}}(x)$, unde numărul natural n şi numărul real x sînt date.

10. Să se calculeze numărul de cifre în scrierea zecimală a numărului natural dat n.

11. Să se verifice dacă numărul natural dat n este putere a lui 2.

12. Se dau numerele naturale m şi n , unde $m < n$. Să se verifice dacă n este o putere a lui m.

13. Să se calculeze $1! + 2! + 3! + ... + n!$ ($n > 1$).

14. Să se verifice dacă numărul dat n este prim.

15. Şirul lui Fibonacci[1)] se defineşte astfel: $f_0 = f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$. Să se afle:
a) al n-lea termen, unde n este dat.
b) primul termen din şir mai mare decît numărul dat k.

16. a) Ce se va afişa după execuţia secvenţei următoare?

```
readln (m);
for i:=1 to m do begin
  inc(i);
  writeln(i);
end;
```

b) Dar dacă schimbăm între ele rîndurile 3 şi 4?

17. Ce va apărea pe ecran în urma execuţiei programului:

a)
```
uses crt;
var i:char;
begin
  clrscr;
  for i:='a' to 'p' do begin
     write(i);
     write(chr(10)); {chr(10) este simbolul 'rind nou'}
  end;
  readkey;
end.
```

b)
```
uses crt;
var i:char;
begin
  clrscr;
  for i:='z' downto 'a' do begin
     write(i);
     write(chr(13)); {chr(13) este simbolul 'inceput de rind'}
  end;
  readkey;
end.
```

18. Se dă numărul natural n. Să se compare sumele S_1 şi S_2, unde:
a) $S_1 = 1^3 + 2^3 + ... + n^3$ şi $S_2 = (1 + 2 + ... + n)^2$;
b) $S_1 = 3(1^2 + 2^2 + ... + n^2)$ şi $S_2 = n^3 + n^2 + (1 + 2 + ... + n)$.

[1)]Leonardo Pisano Fibonacci (1170(80)–1250) – matematician italian.

19. Se dă numărul natural $n > 3$. Să se afişeze triunghiul de numere:

a)
$1\ 2\ \ldots\ n-2\ \ n-1\ \ n$
$1\ 2\ \ldots\ n-2\ \ n-1$
$1\ 2\ \ldots\ n-2$
$\ldots$
$1\ 2$
1

b)
1
$1\ 2$
$1\ 2\ 3$
$\ldots$
$1\ 2\ 3\ \ldots\ n-1$
$1\ 2\ 3\ \ldots\ n-1\ \ n$

c)
1
$1\ 2\ 3$
$1\ 2\ 3\ 4\ 5$
$\ldots$
$1\ 2\ 3\ 4\ \ldots\ 2n\ \ 2n+1$

d)
$1\ 2\ 3\ \ldots\ n-2\ \ n-1\ \ n$
$2\ 3\ \ldots\ n-2\ \ n-1\ \ n$
$\ldots$
$n-2\ \ n-1\ \ n$
$n-1\ \ n$
n

20. Formulaţi enunţul problemei, care se rezolvă cu ajutorul algoritmului:

a)
```
program C1;
var i,n:word;
    S:real;
BEGIN
  write('Introdu n: ');
  readln(n);
  S:=0;
  for i:=1 to n do
     if odd(i) then S:=S+1/i else S:=S-1/i;
  write('Raspuns: S=',S);
END.
```

b)
```
program C2;
var i,n;word;
    s:longint;
BEGIN
  write('Introdu n: ');
  readln(n);
  S:=0;
  i:=1;
  while i<=n do begin
     if odd(i) then S:=S+2*i-1 else S:=S-2*i+1;
     inc(i);
  end;
  write('Raspuns: S=',S);
END.
```

c)
```
program C3;
var n,s:integer;
BEGIN
  write('Introdu n: ');
```

```
  readln(n);
  S:=0;
  repeat
     S:=S+n mod 10;
     n:=n div 10;
  until n=0;
  write('Raspuns: S=',S);
END.
```

d)
```
program C4;
var i,n:word;
    S:longint;
BEGIN
  write('Introdu n: ');
  readln(n);
  S:=0; i:=1;
  while i<=n do begin
     if odd(i) then inc(S,2*i) else dec(S, 2*i);
     inc(i);
  end;
  write('Raspuns: S=',S);
END.
```

e)
```
program C5;
var n:word; cm:byte;
BEGIN
  write('Introdu n: ');
  readln(n);
  cm:=0;
  repeat
     if n mod 10>cm then cm:=n mod 10;
     n:=n div 10;
  until n=0;
  write('Raspuns: CM=', cm);
END.
```

f)
```
program C6;
var i,n:word; k,s:real;
BEGIN
  write('Introdu n: ');
  readln(n);
  S:=0; k:=1;
  for i:=1 to n do begin
     k:=k*1/3;
     if odd(i) then S:S+k else S:=S-k;
  end;
  write('Raspuns: S=',S:=2:2);
END.
```

21. Se dă numărul natural *n*. Să se afişeze divizorii primi ai lui *n*.

22. Să se calculeze:

a) $\sum_{i=1}^{20}\sum_{j=1}^{10}\frac{1}{i+j^2}$; b) $\sum_{i=1}^{30}\sum_{j=1}^{10}\sin(i^2+j^4)$; c) $\sum_{i=1}^{10}\sum_{j=1}^{20}\frac{i-j+1}{i+j}$;

d) $\sum_{i=1}^{20}\sum_{j=1}^{i}\frac{1}{2j+1}$; e) $\sum_{k=1}^{20}\sum_{m=k}^{20}\frac{8+k}{m}$; f) $\sum_{i=1}^{20}\prod_{j=1}^{i}\sin j$;

g) $\sum_{i=1}^{30}\frac{i!}{x^i}$; h) $\prod_{j=1}^{5}\sum_{k=1}^{5}\frac{x}{k}$; i) $\sum_{k=1}^{10}\left(k^3\sum_{i=1}^{15}(k-i)^2\right)$;

j) $\prod_{k=1}^{5}k+\prod_{i=1}^{10}(5-i)^2$.

Observaţie: $\sum_{i=1}^{n}f(i)=f(1)+f(2)+\ldots+f(n)$.

$\prod_{i=1}^{n}f(i)=f(1)\cdot f(2)\cdot\ldots\cdot f(n)$.

23. Să se afle toate numerele prime mai mici decît numărul dat *n*.

24. Să se afle suma cifrelor numărului natural dat *n*.

25. Un număr natural se numeşte *număr perfect* dacă este egal cu suma divizorilor lui, în afară de el însuşi. De exemplu, 6 este număr perfect, deoarece 6 = 1 + 2 + 3. Să se afle numerele perfecte mai mici decît numărul natural dat *n*.

26. Fără a utiliza funcţiile standard (cu excepţia funcţiei modul), să se calculeze cu eroarea dată *e* suma *S* (*x*), apoi să se compare rezultatul, afişînd la ecran şi valoarea funcţiei (aplicînd funcţiile standard) în punctul dat *x*:

a) $S(x)=e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x_n}{n!}+\ldots$;

b) $S(x)=\mathrm{sh}(x)=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}+\ldots$; $\left(\mathrm{sh}\,x=\frac{e^x-e^{-x}}{2}\right)$

c) $S(x)=\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+(-1)^n\frac{x^{2n}}{2n!}+\ldots$;

d) $S(x)=\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\ldots+\frac{(-1)^{n-1}x^n}{n}+\ldots$, unde $|x|<1$;

e) $S(x)=\mathrm{arctg}\,x=x-\frac{x^3}{3}+\frac{x^5}{5}-\ldots+\frac{(-1)^n x^{2n+1}}{2n+1}+\ldots$, unde $|x|\le 1$.

27. Ce va afişa la ecran următorul program?

a)
```
program C1;
uses Crt;
var i,j:byte;
BEGIN
  ClrScr;
```

```
  for i:=1 to 9 do begin
     for j:=1 to i do write (j:3);
     writeln;
  end;
  readkey;
END.
```

b)
```
program C2;
uses Crt;
var i,j:char;
BEGIN
  ClrScr;
  for i:='a' to 'z' do begin
     for j:='a' to i do write (i);
     writeln;
  end;
  readkey;
END.
```

28. Să se calculeze $\sqrt{2+\sqrt{4+...+\sqrt{98+\sqrt{100}}}}$.

29. Să se afişeze la ecran *răsturnatul* numărului natural dat n. De exemplu, răsturnatul numărului 1234 este 4321.

30. Un număr se numeşte *palindrom* dacă el este egal cu răsturnatul său. Să se afle numerele palindroame mai mici decît numărul natural dat n.

31. Se dă numărul natural n. Să se afle media aritmetică a unui şir de n numere date.

32. Se dă numărul natural n. Să se afle media geometrică a unui şir de n numere date.

33. Se dă numărul natural n. Să se afle media armonică a unui şir de n numere date.

34. Se dau numărul real a şi numărul natural n. Să se calculeze $a(a+1)(a+2)...(a+n)$.

35. Să se afişeze la ecran fiecare număr natural mai mare decît 10 şi mai mic decît n, divizibil cu suma cifrelor lui, unde n este număr natural dat mai mare decît 10. De exemplu, pentru $n = 25$, obţinem $12 : (1+2)$, $18 : (1+8)$, $20 : (2+0)$, $24 : (2+4)$.

36. Să se afişeze la ecran fiecare număr natural mai mic decît n, egal cu triplul produsului cifrelor lui, unde n este număr natural dat.

37. Să se afişeze la ecran fiecare număr natural mai mic decît n, avînd pătratul egal cu suma cuburilor cifrelor lui, unde n este număr natural dat.

38. Să se afişeze la ecran fiecare număr natural mai mic decît 1 000, egal cu suma factorialurilor cifrelor lui.

39. Să se afişeze la ecran fiecare număr natural mai mic decît n, cu suma pătratelor cifrelor lui divizibilă cu 17, unde n este număr natural dat. (De exemplu, 14, 28, 29, 35.)

40. Să se afle toate numerele de trei cifre, care se reprezintă ca diferența dintre pătratul numărului format din primele două cifre și pătratul ultimei cifre. De exemplu, $100 = 10^2 - 0^2$.

41. Să se afle toate numerele de trei cifre, pentru care a treia cifră în scrierea zecimală a lor este egală cu media aritmetică a celorlalte două cifre.

42. Să se afle toate numerele de trei cifre, fiecare avînd suma cifrelor egală cu numărul natural dat n.

43. Să se afle toate numerele de k cifre (k este dat), pentru care suma cifrelor nu se schimbă la înmulțirea numărului cu numărul natural dat p, unde $p < 10$.

44. Să se determine dacă numărul natural dat n poate fi reprezentat ca suma pătratelor a două numere naturale.

45. Vom spune că numerele a, b, c sînt numere *pitagoriene* dacă ele pot fi lungimile laturilor unui triunghi dreptunghic. Să se afle toate numerele pitagoriene mai mici decît numărul natural dat n.

46. Să se afle cea mai mică putere a numărului natural dat n, care întrece numărul dat a.

47. Să se afle cea mai mare putere a numărului natural dat n, care nu întrece numărul dat a.

48. Să se verifice dacă pătratul numărului natural dat n conține cifra dată c.

49. Fie l_n lungimea laturii unui poligon regulat cu n laturi înscris în cercul de rază R. Atunci $l_{2n} = \sqrt{2R^2 - R\sqrt{4R^2 - l_n^2}}$, unde l_{2n} este lungimea laturii poligonului cu $2n$ laturi înscris în același cerc. Luînd în considerație faptul că un poligon regulat cu un număr foarte mare de laturi aproximează cercul, să se obțină o aproximare cît mai bună a numărului π.
Indicație. Să se considere $R = 1$.

50. Să se simplifice pînă la o fracție ireductibilă fracția dată $\frac{a}{b}$.

51. Calculați cît mai exact valoarea lui π utilizînd:
a) formula lui François Viète[1]:

$$\frac{2}{\pi} = \sqrt{\frac{1}{2}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2}}}} \cdot \ldots$$

[1] François Viète (1540–1603) – matematician francez.

b) formulele lui Leonard Euler[1]:

$$\frac{\pi^2}{6}=1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+\ldots,$$

$$\frac{\pi^2}{8}=1+\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}+\frac{1}{9^2}+\ldots$$

c) formulele lui John Wallis[2]:

$$\frac{\pi}{2}=\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdot\ldots,$$

$$\frac{4}{\pi}=\frac{3\cdot3\cdot5\cdot5\cdot7\cdot7\cdot9\cdot9}{2\cdot4\cdot4\cdot6\cdot6\cdot8\cdot8\cdot10\cdot10}\cdot\ldots$$

d) formula lui Gottfried Wilhelm von Leibniz[3]:

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots$$

e) formula lui John Machim[4]:

$$\frac{\pi}{4}=4\left(\frac{1}{5}-\frac{1}{3\cdot5^3}+\frac{1}{5\cdot5^5}-\ldots\right)-\left(\frac{1}{239}-\frac{1}{3\cdot239^3}+\frac{1}{5\cdot239^5}-\ldots\right).$$

52. Se dau numerele reale a şi e ($e<0{,}1$). Să se calculeze rădăcina pătrată a lui a cu exactitatea e, fără a utiliza funcţia `sqrt`.

Indicaţie. Şirul recurent al lui Newton[5] $x_0=1$, $x_n=\dfrac{x_{n-1}+\dfrac{a}{x_{n-1}}}{2}$, $n\geq1$, converge către $\sqrt{a}$.

53. Se citeşte de la tastatură o literă care apoi apare în centrul ecranului. Să se deplaseze litera cu ajutorul a 4 taste (sus, jos, stînga, dreapta).

C

54. Să se scrie un algoritm care precizează de cîte ori au fost tastate literele A şi B într-o succesiune de apăsări pe tastele calculatorului.

55. Să se verifice dacă toate cifrele numărului natural dat n ($n<100\,000\,000$) sînt diferite fiecare două.

56. Să se afişeze la ecran a n-a cifră a numărului 12345678910111213141516171819 20...999 (sînt scrise consecutiv toate numerele naturale pozitive mai mici decît 1000), unde n este număr natural dat.

---

[1] Leonard Euler (1707–1783) – matematician elveţian.

[2] John Wallis (1616–1703) – matematician englez.

[3] Gottfried Wilhelm von Leibniz (1646–1716) – matematician german.

[4] John Machim (1685–1751) – matematician englez.

[5] Isaac Newton (1642–1727) – matematician englez.

57. Se dă numărul natural n şi cifra c. De cîte ori apare cifra c în numerotarea paginilor unei cărţi de n pagini?

58. Să se afişeze la ecran a n-a cifră a numărului 112358132134... (sînt scrise consecutiv primele 30 de numere ale şirului Fibonaci), unde n este număr natural dat.

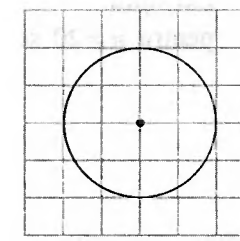
59. Să se scrie un program care va efectua:
a) adunarea a două fracţii date;
b) înmulţirea a două fracţii date.
Rezultatul va fi o fracţie ireductibilă.

60. Se dă numărul natural n. Să se determine al n-lea termen al şirului:
a) 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, ...;
b) 1, 3, 3, 3, 5, 5, 5, 5, 5, ...

61. Se dau numerele naturale m şi n, unde $m < n$. Fie şirul 1, 2, 2, 3, 4, 4, 5, 6, 6, ... Să se afişeze termenii şirului cuprinşi în intervalul $[m, n]$.

62. Să se calculeze, utilizînd schema lui Horner[1)],
$y = x^{10} + 2x^9 + 3x^8 + ... + 10x + 11$, unde x este număr real dat.
Indicaţie. $a_0x^n + ... + a_n = (...(a_0x + a_1)x + ... + a_{n-1})x + a_n$. Se va calcula succesiv $y \leftarrow y \cdot x + a_i$, $i = 0, 1, ..., n$, considerînd iniţial $y = 0$.

63. Pe o reţea de pătrate a fost construit un cerc cu raza de R unităţi ale reţelei. Fiind dat R, să se determine:
a) numărul de pătrate care aparţin în întregime interiorului cercului;
b) numărul de pătrate intersectate de cerc.
De exemplu, pentru $R = 2$ răspunsul va fi:
a) 4; b) 12.



64. Se dă numărul natural n, $n < 100$. Să se determine numărul de zerouri de la sfîrşitul lui $n!$. De exemplu, pentru $n = 10$, unde $10! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 = 3628800$, se va afişa 2.

65. Mihai are un unchi bogat care i-a dăruit în ziua cînd s-a născut un dolar, iar în fiecare an următor el dubla cadoul şi mai adăuga atîţia dolari cîţi ani împlinea Mihai.
a) Să se calculeze cîţi dolari a primit Mihai atunci cînd a împlinit n ani ($n < 20$).
b) La ce vîrstă cadoul lui Mihai a fost mai mare de 100$?

66. Mihai a sădit un măr. În primul an de roadă mărul a avut 8 mere, iar în fiecare an următor roada a crescut cu 50% plus 4 mere.
a) În ce an de roadă mărul a avut nu mai puţin de 200 de mere?
b) Cîte mere a avut mărul în al n-lea an de roadă? (n este dat.)

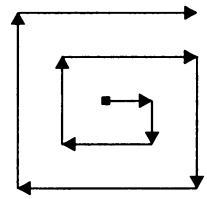
[1)] William George Horner (1768–1837) – matematician englez.

67. Se dă un număr natural n cu cel mult 6 cifre. Să se determine:
a) numărul maxim care se obţine din n eliminînd o cifră.
b) numărul minim care se obţine din n eliminînd o cifră.

68. Să se calculeze:

$$\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{\ddots\,99+\cfrac{1}{101}}}}.$$

69. Se dă numărul natural $n \geq 20$.
Să se afişeze numerele 1, 2, ... n conform schemei.



Exemplu:
pentru $n = 20$ se va afişa:

20	7	8	9	10
19	6	1	2	11
18	5	4	3	12
17	16	15	14	13

4 Tablouri unudimensionale (vectori)

Sugestii teoretice

Declararea unui vector

- *TipInd* este tipul indicilor şi poate fi orice tip ordinal, în afară de tipul integer, adică: char, boolean, enumerare, subdomeniu.

- *TipEl* este tipul elementelor vectorului şi poate fi orice tip simplu sau structurat.

Exemple:

```
type tablou = array [1..30] of char;
var lit_mici, lit_mari: tablou;
    a: array [1..10] of real;
    b: array ['a'..'z'] of integer;
    c: array [boolean] of byte;
```

Probleme rezolvate

❶ Se dă un vector cu n $(1 \le n \le 100)$ componente de tip integer. Să se calculeze numărul de componente nule.

Rezolvare:

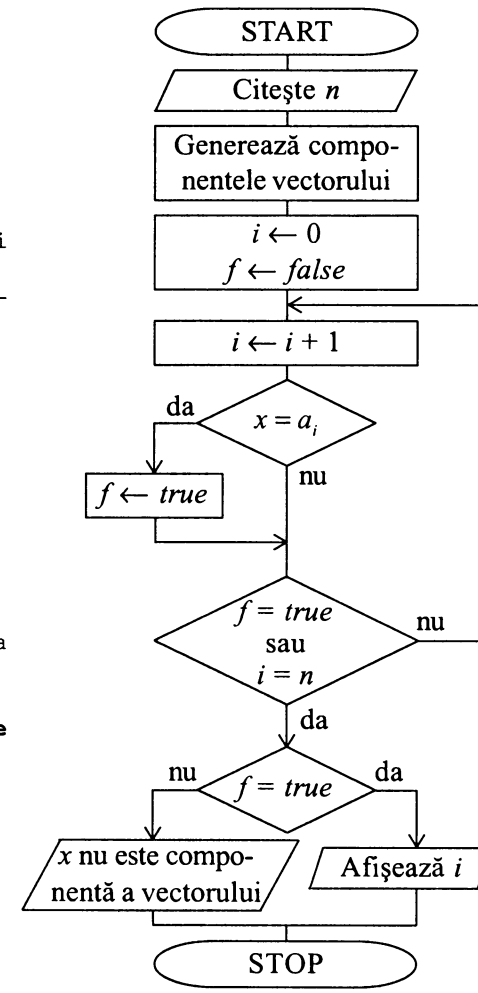
```
program Exemplul1;
uses Crt;
var a:array[1..100] of integer;
    i,z:byte;
BEGIN
   ClrScr;
   wtite('Numarul de componente: ');
   readln(n);
   z:=0;
   for i:=1 to n do begin
      write('Elementul ', i,': ');
      readln(a[i]);
      if a[i]=0 then z:=z+1;
   end;
   {Afisarea vectorului}
```

```
   for i:=1 to n-1 do
      write(a[i], ', ');
   writeln(a[n]);
   {Afisarea numarului de componente nule}
   write('Numarul de componente nule: ', z);
   readkey;
END.
```

❷ Se dă un vector cu n ($1 \le n \le 100$) componente de tip integer şi numărul întreg x. Să se verifice dacă vectorul dat conţine componente egale cu x. În caz afirmativ, să se afişeze poziţia primei componente (egale cu x). De exemplu, pentru vectorul 4, 10, –2, 3, –2, 4, 5 şi $x = -2$ se va afişa „Pozitia 3", iar pentru acelaşi vector şi $x = 7$ se va afişa „Numarul 7 nu este componenta a vectorului".

Rezolvare:

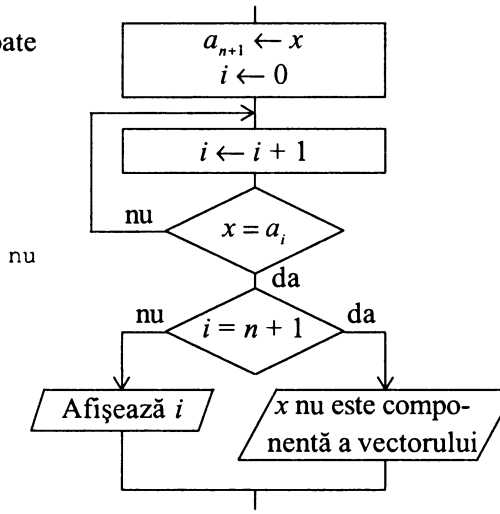
```
program Exemplul2;
var a:array [1..100] of integer;
    n,i,x: integer;
    f:boolean;
BEGIN
   ClrScr;
   write('Numarul de componente: ');
   readln(n);
   Randomize; {Conectarea generatorului
     de numere aleatoare}
   for i:=1 to n do a[i]:=Random(1000)-
     500; {Calculatorul genereaza
     componentele vectorului - numere
     intregi de pe intervalul [-500;
     500)}
   write('Componenta cautata: ');
   readln(x);
   i:=0;
   f:=false;
   repeat
     inc(i);
     f:=(x=a[i]);
     {Daca x=a[i], atunci f ia valoarea
       true};
   until f or (i=n);
   if f then writeln('Pozitia ',i) else
     writeln('Numarul ',x,' nu este
       componenta a vectorului');
   {Afiseaza vectorul}
   for i:=1 to n do
   write(a[i],' ');
   readkey;
END.
```



Observaţie

Secvenţa de căutare a componentei x poate arăta şi astfel:

```
a[n+1]:=x;
i:=0;
repeat
   inc(i);
until x=a[i];
if i=n+1 then write('Numarul ',x,' nu
   este componenta a vectorului')
   else write('Pozitia ', i);
```



❸ Se dă un vector cu n $(1 \le n \le 100)$ componente de tip întreg. Să se ordoneze crescător componentele vectorului.

De exemplu, pentru vectorul 2, 0, 4, –3, 0, –1 se va afişa –3, –1, 0, 0, 2, 4.

Rezolvare:

Se cunosc mai multe metode de sortare a componentelor unui vector. Vom prezenta cîteva.

a) **Sortarea prin selecţie**

Se schimbă locurile primului element şi elementului minim.

Se găseşte elementul minim dintre elementele de pe poziţiile 2, 3, ..., n.

Se schimbă locurile acestui element şi elementului 2.

Se găseşte elementul minim dintre elementele de pe poziţiile 3, 4, ..., n.

Se schimbă locurile acestui element şi elementului 3 etc.

```
program Sort_selectie;
uses Crt;
var a:array[1..100] of integer;
    l,j,i,n: byte;
    k,min:integer;
BEGIN
   ClrScr;
   write('Numarul de componente: ');
   readln(n);
   for i:=1 to n do begin
      write('Elementul ',i,' : ');
      readln(a[i]);
   end;
   writeln('Vectorul initial: ');
   for i:=1 to n-1 do
      write(a[i], ', ');
   writeln(a[n]);
```

```
{Ordonarea componentelor vectorului}
for i:=1 to n-1 do begin
   min:=a[i]; l:=i;
   for j:=i+1 to n do
      if a[j]<min then begin min:=a[j];
         l:=j;
      end;
      k:=a[i]; a[i]:=a[l]; a[l]:=k;
end;
{Afisarea vectorului ordonat}
for i:=1 to n-1 do
   write(a[i], ', ');
writeln(a[n]);
readkey;
END.
```

Executarea algoritmului

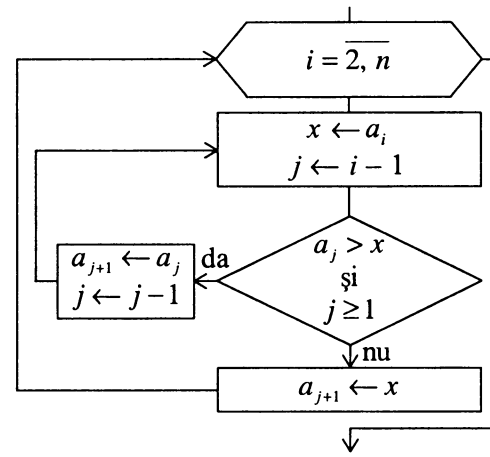
i	Vectorul A: 2, 0, 4, –3, 0, –1
1	**–3**, 0, 4, **2**, 0, –1
2	–3, **–1**, 4, 2, 0, **0**
3	–3, –1, **0**, 2, **4**, 0
4	–3, –1, 0, **0**, 4, **2**
5	–3, –1, 0, 0, **2**, **4**

b) **Sortarea prin inserție**

Considerăm pasul i, cînd toate elementele de la a_1 la a_{i-1} sînt deja ordonate. Vom compara elementul $x = a_i$ cu a_{i-1}, apoi cu a_{i-2} ş.a.m.d. Presupunem că $a_j \le x < a_{j+1}$, unde $j < i$. Atunci x se inserează pe poziția $j + 1$. Elementele $a_{j+1},..., a_{i-1}$ se vor deplasa cu o poziție spre dreapta în timpul comparației.

Prezentăm doar secvența de program pentru ordonarea componentelor:

```
for i:=2 to n do begin
   x:=a[i]; j=i-1;
   while (a[j]>x) and (j>=1) do
   begin
      a[j+1]:=a[j]; dec(j);
   end;
   a[j+1]:=x;
end;
```



Executarea algoritmului

i	Vectorul A: 2, 0, 4, –3, 0, –1
2	0, 2, 4, –3, 0, –1
3	0, 2, 4, –3, 0, –1
4	–3, 0, 2, 4, 0, –1
5	–3, 0, 0, 2, 4, –1
6	–3, –1, 0, 0, 2, 4

c) **Metoda bulelor**

Luînd în considerație că vectorul $A(n)$ este ordonat crescător dacă şi numai dacă $a_i \le a_{i+1}$, $\forall i = \overline{1, n-1}$, vom compara a_i cu a_{i+1}, unde $i = \overline{1, n-1}$. Dacă $a_i > a_{i+1}$, vom schimba locurile acestor elemente.

Observăm că după o parcurgere elementul maximal sigur ajunge ultimul. Dacă se mai execută o parcurgere, maximum 2 ajunge penultimul. În concluzie, pentru siguranță, trebuie să repetăm $n - 1$ parcurgeri.

Pe de altă parte, dacă în urma unei parcurgeri nu a fost efectuată nici o interschimbare (schimbare de locuri), atunci vom trage concluzia că vectorul este deja ordonat. Pentru a şti dacă a fost realizată o interschimbare, vom utiliza o variabilă f de tip logic, care din valoarea iniţială *false* va deveni *true*.

```
Repeat
   f:=false;
   for i:=1 to n-1 do
   if a[i]>a[i+1] then begin
      k:=a[i];
      {k - variabila auxiliara}
      a[i]:=a[i+1];
      a[i+1]:=k;
      f:=true;
   end;
until not f;
```

Executarea algoritmului

Vectorul A: 2, 0, 4, –3, 0, –1	f = *false*
0, 2, –3, 0, –1, 4	*true*
0,–3, 0, –1, 2, 4	*true*
–3, 0, –1, 0, 2, 4	*true*
–3, –1, 0, 0, 2, 4	*true*
–3, –1, 0, 0, 2, 4	*false*

d) **Sortarea prin interschimbare**

Se compară pe rînd primul element a_1 al vectorului cu fiecare din următoarele elemente. Dacă la un moment dat $a_1 > a_i$, atunci vom schimba locurile acestor elemente. Observăm că după o parcurgere elementul minim ajunge pe prima poziţie. Se repetă aceeaşi procedură cu elementul al doilea, apoi cu al treilea, ... , al n–1-lea element (care va fi comparat doar cu ultimul element).

```
for i:=1 to n-1 do
   for j:=i+1 to n do
      if a[i]>a[j] then begin
         k:=a[i];
         {k - variabila auxiliara}
         a[i]:=a[j];
         a[j]:=k;
      end;
```

Executarea algoritmului

i	Vectorul A: 2, 0, 4, –3, 0, –1
1	–3, 2, 4, 0, 0, –1
2	–3, –1, 4, 2, 0, 0
3	–3, –1, 0, 4, 2, 0
4	–3, –1, 0, 0, 4, 2
5	–3, –1, 0, 0, 2, 4

❹ De la tastatură se citeşte un text pînă la apăsarea tastei Enter. Să se numere de cîte ori se repetă în text fiecare literă a alfabetului latin.

Rezolvare:

```
program Litere1;
uses Crt;
var a:array[char] of integer;
    c:char;
BEGIN
   ClrScr;
   read(c);
   while c<>chr(10) do begin
      inc(a[Upcase(c)]);
      read(c);
   end;
```

```
   for c:='A' to 'Z' do
   if a[c]<>0 then writeln('Litera ',c,' se repeta de ',a[c],' ori');
   readkey;
END.
```

Observaţie

La apăsarea tastei Enter se citesc simbolurile „rînd nou" (chr(10)) şi „început de rînd" (chr(13)).

❺ De la tastatură se citeşte un text pînă la apăsarea tastei Enter. Literele mici vor cădea cîte una pe rîndul 24 al ecranului.

Rezolvare:

```
program Litere2;
uses Crt;
var a:array[1..100] of char;
    c:char;
    i,j,t:integer;
BEGIN
   ClrScr;
   i:=0;
   writeln('Scrie un text:');
   read(c);
   while c<>chr(10) do begin
      inc(i);
      a[i]:=c;
      read(c);
   end;
   for j:=1 to i do
      if Upcase(a[j])<>a[j] then begin
         for t:=2 to 24 do begin
            gotoXY(j,t);
            write(' ');
            gotoxy(j,t+1);
            write(a[j]);
            delay(300);
         end;
      end;
   readkey;
END.
```

❻ Se dă un vector cu n ($n < 100$) componente întregi, astfel încît $a_1 \le a_2 \le ... \le a_n$. Să se determine lungimea celui mai mare subşir de componente consecutive egale. De exemplu, pentru vectorul –2, –2, 0, 0, 0, 2, 3, 3, 3, 3, 5, 7, 7, 7 se va afişa valoarea 4.

Rezolvare:

```
program Vector6;
uses Crt;
var v:array[1..50] of integer;
    i,n,t:integer;
BEGIN
   ClrScr;
```

```
  write('Introdu dimensiunea vectorului: ');
  readln(n);
  for i:=1 to n do begin
    write('v[',i,']: ');
    readln(v[i]);
  end;
  writeln('-----------');
  for i:=1 to n do
    write(v[i],' ');
  writeln;
  i:=1;
  t:=1;
  while i<n-1 do begin
    if v[i-t+1]=v[i+1] then inc(t);
    inc(i);
  end;
  writeln('Numarul maximal de componente egale: ',t);
  readkey;
END.
```

❼ Se dă un vector cu n ($n < 100$) componente întregi. Să se afişeze poziţiile componentelor, care sînt cel mai apropiate de media aritmetică a componentelor vectorului. De exemplu, pentru vectorul 1, 2, 3, 4, 5, –5, –4, –3, –2, –1 se vor afişa poziţiile 1 şi 10.

Rezolvare:

```
program Vector7;
uses Crt;
var v:array[1..50] of integer;
    i,n,s:integer;
    media,min:real;
BEGIN
  ClrScr;
  write('Introdu dimensiunea vectorului: ');
  readln(n);
  for i:=1 to n do begin
     write('v[',i,']: ');
     readln(v[i]);
  end;
  writeln('-------------');
  S:=0;
  for i:=1 to n do begin
     S:=S+v[i];
     write(v[i],' ');
  end;
  writeln;
  media:=S/n;
  min:=abs(media-v[1]);
  for i:=2 to n do
     if abs(media-v[i])<min then min:=abs(media-v[i]);
  writeln('Media aritmetica este egala cu ',media:2:2);
  writeln('Componentele cele mai apropiate de medie se afla pe
     pozitiile: ');
```

```
    for i:=1 to n do
      if abs(media-v[i])=min then write(i,' ');
    readkey;
END.
```

Probleme propuse

A

1. Se dă un vector cu n $(10 \le n \le 100)$ componente întregi.
   a) Să se afişeze la ecran componenta: a treia; a patra; a noua.
   b) Să se calculeze suma componentelor a doua, a treia şi a opta.
   c) Să se mărească cu 5 prima şi ultima componentă.
   d) Să se micşoreze cu 10 componenta (eventual cele 2 componente) din mijloc.

2. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se calculeze:
   a) suma componentelor lui; b) produsul componentelor lui.

3. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se calculeze numărul componentelor:
   a) negative;
   b) pare;
   c) nenule;
   d) pozitive divizibile cu 3 şi cu 5;
   e) divizibile cu cel puţin unul dintre numerele 7, 9, 11;
   f) al căror modul este mai mare decît 3.

4. *Şirul lui Fibonacci.* Să se construiască un vector cu n $(1 \le n \le 100)$ componente ce vor satisface relaţiile: $a_1 = 1$, $a_2 = 1$, $a_i = a_{i-2} + a_{i-1}$ $(i \ge 3)$.

5. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Dacă vectorul conţine componente egale cu 0, să se afişeze poziţia primei atare componente, în caz contraz să se afişeze mesajul „Vectorul nu conţine 0".

6. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se ordoneze componentele vectorului în ordine:
   a) crescătoare; b) descrescătoare.

7. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi şi numărul natural t $(1 < t < n)$. Să se ordoneze primele t componente crescător, iar celelalte – descrescător.

8. Se dă un vector cu n $(1 \le n \le 100)$ componente numere naturale. Să se afişeze la ecran componentele prime (divizibile doar cu 1 şi cu ele înseşi).

9. Se dă un vector cu n $(1 \le n \le 100)$ componente numere reale. Să se verifice dacă vectorul conține cel puțin o pereche de componente alăturate, care sînt:
a) numere opuse (suma lor este 0);
b) numere inverse (produsul lor este 1).

10. Se dă un vector cu n $(1 \le n \le 100)$ componente numere naturale. Să se calculeze produsul primelor componente a căror sumă nu întrece numărul natural dat m. (Dacă se va aduna următoarea componentă, suma va fi mai mare sau egală cu m.)

11. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se determine:
a) componenta maximală și toate pozițiile ei;
b) componenta minimală și toate pozițiile ei;
c) componenta maximală ce nu întrece numărul întreg dat m;
d) componenta minimală ce nu întrece numărul întreg dat m;
e) componenta maximală negativă;
f) componenta minimală pozitivă.

12. Se dă un vector cu n $(1 \le n \le 100)$ componente întregi. Să se determine:
a) componenta maximală dintre cele pare;
b) componenta minimală dintre cele impare;
c) componenta maximală dintre cele prime;
d) componenta minimală dintre cele compuse.

13. Se dă un vector A cu $2n$ $(1 \le n \le 50)$ componente numere întregi. Să se verifice dacă vectorul A este simetric față de mijlocul lui, adică au loc relațiile: $a_1 = a_{2n}$, $a_2 = a_{2n-1}$, $a_3 = a_{2n-2}$, ..., $a_n = a_{n+1}$.

14. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi.
a) Să se mărească cu 2 toate componentele ce urmează după prima componentă pozitivă.
b) Să se micșoreze cu 2 toate componentele ce preced prima componentă negativă.
c) Să se mărească de 2 ori toate componentele ce urmează după ultima componentă negativă.
d) Să se micșoreze de 2 ori toate componentele ce preced ultima componentă pozitivă.

15. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se verifice dacă vectorul conține:
a) 2 componente vecine egale cu 0;
b) 3 componente vecine de același semn.

16. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi. Să se calculeze media aritmetică a componentelor cu indici pari și media aritmetică a componentelor cu indici impari.

17. Se dă un vector cu n $(1 \le n \le 100)$ componente numere întregi.

a) Să se substituie cu 0 fiecare a treia componentă, începînd cu prima.

b) Să se substituie cu opusa sa fiecare a patra componentă, începînd cu a doua.

c) Să se substituie cu componenta maximală fiecare a doua componentă, începînd cu prima.

d) Să se substituie fiecare a cincea componentă cu cea minimală.

18. Să se formuleze enunțul problemei care se rezolvă cu ajutorul algoritmului:

a)
```
program V1;
uses Crt;
var a,b,c:array[1..100] of integer;
    i,j,n,k:byte;
    este:boolean;
BEGIN
  readln(n);
  for i:=1 to n do begin
     readln(a[i]);
     readln(b[i]);
  end;
  for i:=1 to n do begin
     este:=false
     for j:=1 to n do
        if a[i]=b[j] then este:=true;
     if not este then begin
        inc(k);
        c[k]:=a[i];
     end;
  end;
  for i:=1 to k do
     write(c[i]:3);
  readkey;
END.
```

b)
```
program V2;
uses Crt;
var a:array[1..100] of integer;
    i,n,m,k:integer;
BEGIN
  readln(n);
  m:=-MaxInt; k:=0;
  for i:=1 to n do begin
     if a[i]=m then k:=succ(k);
     if a[i]>m then begin
        m:=a[i];
        k:=1;
     end;
  end;
  write('Raspuns: ',k);
  readkey;
END.
```

c)
```
program V3;
uses Crt;
var a:array[1..100] of integer;
    i,n,m:integer;
BEGIN
  readln(n);
  for i:=1 to n do
     readln(a[i]);
  m:=a[1];
  for i:=2 to n do
     if a[i]<m then m:=a[i];
  for i:=1 to n do begin
     a[i]:=a[i]-m;
     write(a[i]:3);
  end;
  readkey;
END.
```

d)
```
program V4;
uses Crt;
var a:array[1..100] of integer;
    i,n,m:integer;
BEGIN
  readln(n);
  for i:=1 to n do
     readln(a[i]);
  for i:=1 to n div 2 do begin
     m:=a[i];
     a[i]:=a[n+1-i];
     a[n+1-i]:=m;
  end;
  for i:=1 to n do
     write(a[i]:3);
  readkey;
END.
```

e)
```
program V5;
uses Crt;
var a:array[1..100] of integer;
    i,j,n,m:integer;
    diferit:boolean;
BEGIN
  readln(n);
  for i:=1 to n do
     readln(a[i]);
  m:=0;
  for i:=1 to n do begin
     j:=i+1; diferit:=true;
     while (j<=n) and diferit do
        if a[i]<>a[j] then inc(j) else diferit:=false;
     if diferit then m:=m+1;
  end;
```

```
  write('Raspuns: ',m);
  readkey;
END.
```

B

19. De la tastatură se citeşte un text pînă la apăsarea tastei Enter. Să se numere:
a) de cîte ori se repetă în text fiecare dintre literele „a" şi „b";
b) de cîte ori se repetă în text silaba „oa".

20. Se dă un vector cu n ($n < 100$) componente întregi. Să se determine maximum2 – componenta cea mai mare dintre toate componentele în afară de componenta maximală.

21. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se afişeze componenta care apare de cele mai multe ori în vector. Dacă există mai multe astfel de componente, să se afişeze toate.

22. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se calculeze numărul componentelor diferite. De exemplu, pentru vectorul 14, –2, 3, 14, 3, –2, 5 se va afişa 4.

23. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se determine cea mai lungă secvenţă de zerouri consecutive (la ecran se va afişa lungimea).

24. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se determine cea mai lungă secvenţă de componente consecutive ordonate descrescător (la ecran se va afişa secvenţa).

25. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se schimbe între ele locurile componentelor maximale şi minimale. De exemplu, pentru vectorul 1, –7, 4, –5, –7, 2, 0, 4 se va afişa: 1, 4, –7, –5, 4, 2, 0, –7.

26. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi. Să se verifice dacă vectorul reprezintă o mulţime. De exemplu, pentru vectorul –3, 4, 1, 0, 5 se va afişa mesajul „Vectorul reprezinta o multime", iar pentru vectorul 1, 0, –7, 3, 2, –7 – mesajul „Vectorul nu reprezinta o multime".

27. Se dă un vector cu n ($1 \le n \le 100$) componente numere întregi.
a) Să se permute circular cu o poziţie la dreapta componentele vectorului (de exemplu, 1, 2, 3, 4, 5 devine 5, 1, 2, 3, 4).
b) Să se permute circular cu o poziţie la stînga componentele vectorului (de exemplu, 1, 2, 3, 4, 5 devine 2, 3, 4, 5, 1).

28. Se dă un vector cu n ($1 \le n \le 50$) componente de tip integer. Să se insereze între fiecare două componente:
a) suma lor (de exemplu, pentru –1, 2, 5, 0, 2 se va obţine –1, 1, 2, 7, 5, 5, 0, 2, 2);

b) suma celorlalte componente (de exemplu pentru –1, 2, 5, 0, 2 se va obține –1, 7, 2, 2, 5, 2, 0, 0, 2).

C

29. *Ciurul lui Eratostene*[1]. Se dă numărul natural n. Să se determine numerele prime mai mici sau egale cu n prin următoarea metodă: se substituie cu 0 numerele divizibile cu 2, apoi cele divizibile cu 3, apoi cele divizibile cu 5 ş.a.m.d.

30. Se dă un tablou cu n $(1 \leq n \leq 100)$ componente întregi. Să se determine toate tripletele:
a) crescătoare;
b) care nu sînt nici crescătoare, nici descrescătoare.

31. Se dau 2 vectori de numere întregi, fiecare reprezentînd o mulțime. (La citirea componentelor algoritmul va urmări ca vectorul să fie mulțime). Să se determine:
a) reuniunea mulțimilor;
b) intersecția mulțimilor;
c) diferența mulțimilor.

32. Se dă un vector cu n $(1 \leq n \leq 100)$ componente naturale şi numărul natural s. Să se determine numărul minim de componente cu suma nu mai mică decît s. Să se afişeze aceste componente.

33. Se dau 2 vectori cu n $(1 \leq n \leq 100)$ componente întregi. Să se determine subşirul de componente consecutive şi de lungime maximă conținut în ambii vectori. De exemplu, pentru şirurile
4, –3, 2, 1, 5, 9, 0, 4, –3, 6 şi 2, –3, 2, 0, 5, 9, 0, 4, 2, 6 se va afişa 5, 9, 0, 4.

34. Se dau 2 vectori cu n, respectiv m componente numere de o cifră $(n, m \leq 100)$. Fiecare vector reprezintă un număr natural. Să se realizeze un algoritm care va efectua:
a) adunarea numerelor;
b) scăderea numerelor.
Rezultatul se va memoriza într-un alt vector.

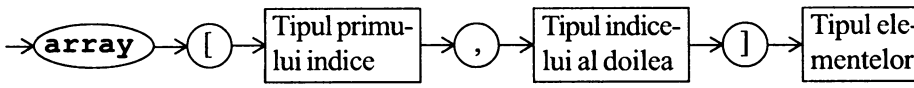
35. Două numere naturale se numesc *gemene*, dacă ele sînt prime şi diferența lor este 2. De exemplu, perechile (3, 5), (5, 7), (11, 13) sînt numere gemene.
Să se determine numerele gemene mai mici decît numărul natural dat n.
Indicație. Se aplică *ciurul lui Eratostene*. (vezi probl. 29)

[1] Eratostene din Alexandria (276 – 196 î. Hr.) – matematician, astronom, filosof antic.

5 Tablouri bidimensionale

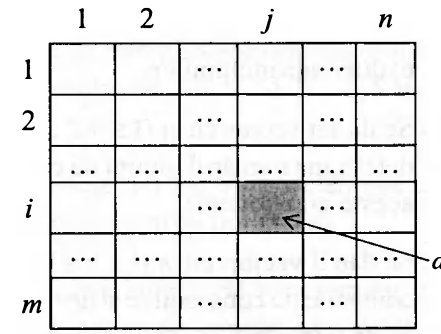
Sugestii teoretice

Declararea tipului tablou bidimensional

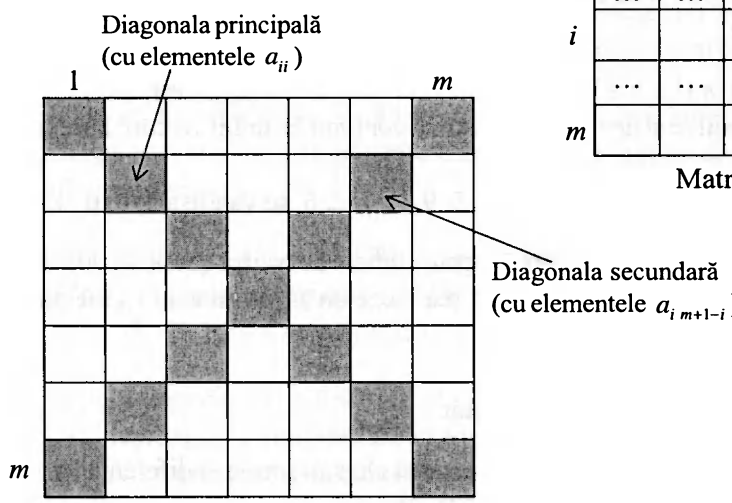


De exemplu, matricea $A(m, n)$ cu m linii şi n coloane, ale cărei elemente sînt numere întregi, poate fi declarată astfel:

```
const m=10, n=15;
var A:array [1..m, 1..n] of integer;
```



Matricea $A(m, n)$



Matricea $A(m, m)$

Probleme rezolvate

❶ Să se scrie un program care va calcula pentru fiecare coloană a unei matrice de elemente întregi suma elementelor.

Astfel, pentru matricea $\begin{pmatrix} -1 & 4 & 3 & 1 \\ 5 & 6 & 0 & 11 \\ -2 & 0 & -7 & 4 \end{pmatrix}$ se va afişa:

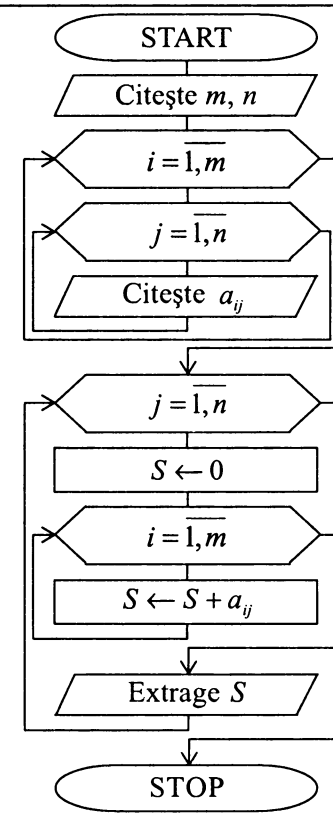
coloana 1: 2
coloana 2: 10
coloana 3: –4
coloana 4: 16

Rezolvare:

```
program Matr1;
uses Crt;
var a:array[1..15,1..15] of integer;
    i,j,m,n,s:integer;
BEGIN
   ClrScr;
   write('Numarul de linii: ');
   readln(m);
   write('Numarul de coloane: ');
   readln(n);
   for i:=1 to m do                    } citirea matricei de la tastatură
      for j:=1 to n do begin
         write('A[',i,', ',j,']=');
         readln(A[i,j]);
      end;
   ClrScr;
   writeln('Matricea A');              } afişarea la ecran a matricei
   for i:=1 to m do begin
   {Extragerea matricei pe linii}
      for j:=1 to n do
         write(A[i,j],' ');
         {extragerea liniei i}
      writeln;
      {transferarea cursorului in
         linie noua}
   end;
   for j:=1 to n do begin
      S:=0;
      for i:=1 to m do S:=S+a[i,j];
      writeln('Coloana ',j,': ', S);
   end;
   readkey;
END.
```

Executarea algoritmului

j	i	S
1		0
	1	$0+(-1)=-1$
	2	$-1+5=4$
	3	$4+(-2)=2$
2		0
	1	$0+4=4$
	2	$4+6=10$
	3	$10+0=10$
...		



❷ Să se se scrie un algoritm Pascal care va determina într-o matrice de numere întregi elementul minimal şi toate poziţiile (indicii) lui.

Astfel, fiind dată matricea $\begin{pmatrix} -4 & 6 & 0 & \boxed{-5} \\ 2 & 11 & 4 & -4 \\ 2 & \boxed{-5} & 6 & \boxed{-5} \end{pmatrix}$, se va afişa: min = −5
Pozitiile: 1, 4
3, 2
3, 4

Rezolvare:

```
program Matr2;
uses Crt;
var a:array[1..20,1..20] of integer;
    m,n,i,j,min:integer;
BEGIN
   ClrScr;
   write('Numarul de linii ------->');
   readln(m);
   write('Numarul de coloane ------->');
   readln(n);
   for i:=1 to m do
      for j:=1 to n do begin
         writeln('A[',i,', ',j,']=');
         readln(a[i,j]);
      end;
   ClrScr;
   for i:=1 to m do begin
      for j:=1 to n do
         write(a[i,j],' ');
      writeln;
   end;
```

determinarea elementului minimal

```
   min:=a[1, 1];
   for i:=1 to m do
      for j:=1 to n do
         if a[i,j]<min then min:=a[i,j];
   writeln('min= ',min);
   writeln('Pozitiile:');
   for i:=1 to m do
      for j:=1 to n do
         if a[i,j]=min then
            writeln(i,', ',j);
   readkey;
END.
```

❸ Se dă numărul natural n. Să se construiască matricea $A(n, n)$ de forma:

1	2	3	...	$n-2$	$n-1$	n
2	1	2	...	$n-3$	$n-2$	$n-1$
3	2	1	...	$n-4$	$n-3$	$n-2$
...						
$n-1$	$n-2$	$n-3$	...	2	1	2
n	$n-1$	$n-2$	...	3	2	1

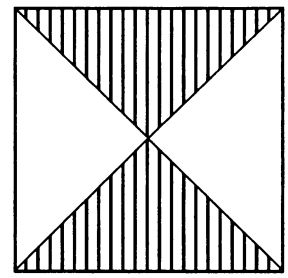
Rezolvare:

Observăm că matricea este simetrică față de diagionala principală. Prin urmare, $a_{ij} = a_{ji}$.

```
program Matr3;
uses Crt;
var a:array[1..30,1..30] of byte;
    i,j,n:byte;
```

```
BEGIN
   ClrScr;
   write('Introdu dimensiunea matricei: ');
   readln(n);
   for i:=1 to n do
      for j:=i to n do begin
         a[i,j]:=j-i+1;
         a[j,i]:=a[i,j];
      end;
   {Afisarea matricei}
   for i:=1 to n do begin
      for j:=1 to n do
         write(a[i,j]:3);
      writeln;
   end;
   readkey;
END.
```

❹ Se dă matricea $A(n, n)$ de numere întregi, unde n este număr natural impar. Să se calculeze suma elementelor domeniului haşurat.



De exemplu, pentru $n = 5$

şi matricea:

2	–2	1	0	5
1	0	7	4	–6
1	6	9	3	1
4	8	–5	1	0
1	2	3	4	5

suma va fi

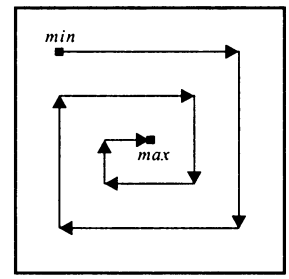
$$\begin{aligned} &2 - 2 + 1 + 0 + 5 + \\ &\quad + 0 + 7 + 4 + \\ &\qquad + 9 + \\ &\quad + 8 - 5 + 1 + \\ &+1 + 2 + 3 + 4 + 5 \\ \hline &= 45 \end{aligned}$$

Rezolvare:

```
program Matr4;
uses Crt;
var a:array[1..30,1..30] of integer;
    n,i,j,s,k:integer;
BEGIN
   ClrScr;
   repeat
      write('Introdu dimensiunea matricei: ');
      readln(n);
```

```
   until odd(n);
   for i:=1 to n do
      for j:=1 to n do begin
         write('a[',i,',',j,']: ');
         readln(a[i,j]);
      end;
   {Afisarea matricei}
   for i:=1 to n do begin
      for j:=1 to n do
         write(a[i,j]:3);
      writeln;
   end;
   k:=n div 2;
   s:=0;
   for i:=1 to k do
      {sumam elementele liniei i si liniei n+1-i}
      for j:=i to n+1-i do
         s:=s+a[i,j]+a[n+1-i,j];
   s:=s+a[k+1,k+1]; {adaugam elementul din centrul matricei}
   writeln('Suma: ',S);
   readkey;
END.
```

❺ Se dă matricea $A(n, n)$ de numere întregi. Să se ordoneze componentele matricei A conform schemei



De exemplu, pentru matricea

1	3	2	5
0	–4	5	4
7	6	9	8
11	2	1	6

se va afişa

–4	0	1	1
6	7	8	2
6	11	9	2
5	5	4	3

Rezolvare:

Folosim un vector $B(n^2)$ în care trecem componentele matricei A. Ordonăm crescător componentele vectorului, apoi le plasăm în matrice parcurgînd-o conform schemei.

```
program Matr5;
uses Crt;
var     a:array[1..10,1..10] of integer;
   b:array[1..100] of integer;
   i,j,k,n,t:integer;
   f:boolean;
BEGIN
   ClrScr;
   write('Introdu dimensiunea matricei: ');
   readln(n);
   for i:=1 to n do
      for j:=1 to n do begin
         write('a[',i,',',j,']: ');
         readln(a[i,j]);
      end;
   writeln('---------------');
   writeln('Matricea initiala');
   writeln('---------------');
   for i:=1 to n do begin
      for j:=1 to n do
         write(a[i,j]:4);
      writeln;
   end;
   k:=0;
   for i:=1 to n do
      for j:=1 to n do begin
         inc(k);
         b[k]:=a[i,j];
      end;
   {Ordonarea vectorului B prin metoda bulelor}
   repeat
      f:=false;
      for i:=1 to n*n-1 do
         if b[i]>b[i+1] then begin
            t:=b[i];
            b[i]:=b[i+1];
            b[i+1]:=t;
            f:=true;
         end;
   until not f;
   {Plasarea elementelor inapoi in matrice}
   k:=0;
   t:=0;
   while k<n*n do begin
      for i:=1+t to n-t do begin {la dreapta}
         inc(k);
         a[t+1,i]:=b[k];
      end;
      for i:=2+t to n-t do begin {in jos}
         inc(k);
         a[i,n-t]:=b[k];
      end;
      for i:=n-t-1 downto 1+t do begin {la stinga}
```

```
        inc(k);
        a[n-t,i]:=b[k];
      end;
      for i:=n-t-1 downto 2+t do begin {in sus}
        inc(k);
        a[i,1+t]:=b[k];
      end;
      inc(t);
    end;
    writeln('----------------');
    writeln('Matricea ordonata');
    writeln('----------------');
    for i:=1 to n do begin
      for j:=1 to n do
        write(a[i,j]:4);
      writeln;
    end;
    readkey;
END.
```

Probleme propuse

A

1. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afişeze la ecran:
   a) suma elementelor matricei;
   b) suma elementelor pozitive;
   c) numărul elementelor negative.

2. Se dă matricea $A(m, n)$, ale cărei elemente sînt numere întregi, şi numărul natural t, $t \le m$, $t \le n$. Să se afişeze la ecran elementele:
   a) liniei t;
   b) coloanei t.

3. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle cel mai mare element al matricei şi numărul liniei şi coloanei corespunzătoare lui.

4. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle cel mai mic element al matricei şi numărul liniei şi coloanei corespunzătoare lui.

5. Pentru fiecare linie a matricei date $A(m, n)$, ale cărei elemente sînt numere întregi, să se calculeze suma şi numărul de elemente pozitive.

6. Pentru fiecare coloană a matricei date $A(m, n)$ ale cărei elemente sînt numere întregi, să se calculeze suma şi numărul de elemente negative.

7. Pentru fiecare linie a matricei date $A(m, n)$, ale cărei elemente sînt numere întregi, să se calculeze media aritmetică a elementelor pozitive.

8. Pentru matricea dată $A(m, n)$, ale cărei elemente sînt numere întregi, să se verifice dacă suma elementelor ei este număr par.

9. Pentru matricea dată $A(m, n)$, ale cărei elemente sînt numere întregi, să se verifice dacă suma elementelor pozitive ale ei este număr impar.

10. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle numărul de elemente ale matricei, mai mari decît:
a) media aritmetică a elementelor ei;
b) media geometrică a elementelor ei;
c) media armonică a elementelor ei.

11. Se dă matricea $A(m, n)$ cu elemente numere naturale. Să se calculeze suma şi produsul elementelor care:
a) se divid cu 3;
b) fiind împărţite la 2, au cîtul număr par;
c) se divid cu 3 sau cu 2;
d) se divid cu 3, dar nu se divid cu 6.

12. Se dă matricea $A(m, n)$ şi numerele naturale p şi q, unde p şi q sînt mai mici sau egale cu m şi n. Să se schimbe locurile:
a) coloanelor p şi q;
b) liniilor p şi q.

13. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se construiască vectorul $X(n)$, unde elementul x_i este egal cu:
a) suma elementelor pozitive din coloana i a matricei A;
b) media aritmetică a elementelor din coloana i a matricei A;
c) cel mai mare element din coloana i a matricei A.

14. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se schimbe locurile elementelor maximal şi minimal. (Se va cerceta şi cazul cînd matricea are mai multe elemente minimale şi maximale.)

15. Fiind dat numărul natural n, să se construiască matricea $A(n, n)$ de forma:

a)
$$\begin{matrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & n \end{matrix}$$

b)
$$\begin{matrix} n & n-1 & n-2 & \dots & 1 \\ 1 & n & n-1 & \dots & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots & n \end{matrix}$$

c)
$$\begin{matrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 2 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 2 & \dots & n-1 & n \end{matrix}$$

16. Se dau matricele $A(m, n)$ şi $B(m, n)$ cu elemente numere întregi. Să se calculeze numărul de perechi (a_{ij}, b_{ij}), pentru care:
a) $a_{ij} < b_{ij}$; b) $|a_{ij}| = |b_{ij}|$; c) $|a_{ij}| > b_{ij}$.

17. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afişeze la ecran elementele diagonalei principale.

18. Să se construiască matricea $A(m, n)$ ale cărei elemente se determină din egalitatea $a_{ij} = i + j$.

19. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle cel mai mare element al diagonalei principale şi să se afişeze la ecran linia care conţine acest element.

20. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle cel mai mic element al diagonalei principale şi să se afişeze la ecran coloana care conţine acest element.

21. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se calculeze suma şi produsul elementelor negative impare care satisfac relaţia $|a_{ij}| < i$.

22. Se dau matricele $A(n, n)$, $B(n, n)$ şi $C(n, n)$ cu elemente numere reale. Să se construiască matricea $X(n, n)$, unde $x_{ij} = \max\{a_{ij}, b_{ij}, c_{ij}\}$.

23. Se dă matricea $A(n, n)$, ale cărei elemente sînt numere întregi, şi numărul întreg p, unde p este mai mic sau egal cu n. Să se schimbe locurile coloanei p şi liniei p.

24. Se dă matricea $A(n, n)$ cu elemente numere întregi. Să se afle suma elementelor situate în liniile cu elemente negative în diagonala principală.

25. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se afle:
a) numărul elementelor nenule ale fiecărei linii a matricei;
b) numărul elementelor pozitive ale fiecărei coloane a matricei;
c) numărul elementelor impare ale fiecărei diagonale a matricei.

26. Să se scrie un program care va efectua:
a) adunarea a două matrice;
b) înmulţirea a două matrice.

27. Se dă matricea $A(n, n)$ cu elemente numere întregi. Componentele matricei $B(n, n)$ se calculează cu formula

a) $b_{ij} = \begin{cases} a_{ij}, \text{ dacă } a_{ij} \geq 0 \\ 1, \text{ dacă } a_{ij} < 0; \end{cases}$ b) $b_{ij} = \begin{cases} 0, \text{ dacă } a_{ij} \text{ este par} \\ a_{ij} + 2, \text{ dacă } a_{ij} \text{ este impar.} \end{cases}$

Să se scrie un program care va efectua adunarea matricelor $A(n, n)$ şi $B(n, n)$.

28. Se consideră m fete şi n băieţi şi matricea $A(m, n)$ formată din zerouri şi unităţi:

$$a_{ij} = \begin{cases} 1, \text{ dacă fata } i \text{ simpatizează băiatul } j \\ 0, \text{ în caz contrar.} \end{cases}$$

Să se determine băiatul (eventual băieţii) care este simpatizat de cele mai multe fete.

29. În matricea $A(n, n)$ au fost înregistrate rezultatele meciurilor jucate de n echipe de fotbal (fiecare 2 echipe au jucat o dată între ele). Înfrîngerile au fost punctate cu 0 puncte, egalităţile – cu 1 punct, iar victoriile – cu 2 puncte. Astfel,

$$a_{ij} = \begin{cases} 0, \text{ dacă în jocul } (i, j) \text{ a învins echipa } j \\ 1, \text{ dacă în jocul } (i, j) \text{ a fost egalitate} \\ 2, \text{ dacă în jocul } (i, j) \text{ a învins echipa } i. \end{cases}$$

Evident, $a_{ji} = 2 - a_{ij}$. Considerăm $a_{ii} = 0$. Să se afişeze numerele echipelor în ordine descrescătoare a punctajului total.

30. Se dă matricea $A(n, n)$, ale cărei elemente sînt numere întregi, şi numerele p şi q mai mici sau egale cu n. Să se obţină matricea $B(n - 1, n - 1)$, eliminînd linia p şi coloana q a matricei A.

31. Se dă matricea $A(m, n)$ cu elemente numere întregi distincte.

a) În fiecare linie se alege elementul maximal, apoi din numere alese se găseşte elementul minimal. Să se afişeze acest element şi poziţia lui în matrice.

b) În fiecare coloană se alege elementul minimal, apoi din numerele alese se găseşte elementul maximal. Să se afişeze la ecran acest element şi poziţia lui în matrice.

32. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se schimbe locurile liniilor astfel încît sumele elementelor fiecărei linii să formeze un şir crescător.

33. Se dă matricea $A(n, n)$ cu elemente numere întregi. Să se calculeze suma elementelor situate mai sus de diagonala principală şi suma elementelor situate mai jos de ea.

34. Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se schimbe locurile liniilor şi coloanelor, astfel încît elementul minimal să fie situat în colţul:

a) stînga-sus;

b) stînga-jos;

c) dreapta-sus;

d) dreapta-jos.

35. Se dă matricea $A(m, m)$ cu elemente numere întregi. Să se obţină matricea $B(m, m)$:

$$B=\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} & \dots & a_{1m} \\ 0 & 0 & a_{23} & a_{24} & \dots & a_{2m} \\ 0 & 0 & 0 & a_{34} & \dots & a_{3m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots & a_{m-1\ m} \\ 0 & 0 & 0 & \dots & \dots & 0 \end{pmatrix}$$

Să se afişeze la ecran ambele matrice.

36. Se consideră matricea $A(m, n)$ de numere întregi. Să se explice ce problemă se rezolvă în următoarea secvenţă de program:

a)
```
for j:=1 to n do begin
  t:=a[2,j];
  a[2,j]:=a[4,j];
  a[4,j]:=t;
end;
```

b)
```
for i:=1 to m do begin
  t:=a[i,n-1];
  a[i,n-1]:=a[i,n];
  a[i,n]:=t;
end;
```

c)
```
for i:=1 to m do begin
  m:=a[i,1];
  for j:=2 to n do
     if a[i,j]<m then m:=a[i,j];
  write(m,' ');
end;
```

d)
```
for j:=1 to n do begin
  m:=a[1,j]; l:=1;
  for i:=2 to m do
     if a[i,j]>m then begin
        m:=a[i,j];
        l:=i;
     end;
  write(l,' ');
end;
```

37. Se consideră matricea $A(m, m)$ de numere întregi. Să se explice ce problemă se rezolvă în următoarea secvenţă de program:

a)
```
s:=0;
for i:=2 to m do
  for j:=1 to i-1 do
     s:=s+a[i,j];
write(s);
```

b)
```
s:=0;
for i:=1 to m-1 do
  for j:=i+1 to m do
     s:=s+a[i,j];
write(s);
```

c)
```
s:=0;
for i:=1 to m-1 do
  for j:=1 to m-i do
     s:=s+a[i,j];
write(s);
```

d)
```
for i:=2 to m do
  for j:=m downto m-i+2 do
     s:=s+a[i,j];
write(s);
```

e)
```
s:=0;
for i:=1 to m do
  s:=s+a[i,i]+a[i,m-i+1];
k:=m div 2+1;
if m mod 2<>0 then s:=s-a[k,k];
write(s);
```

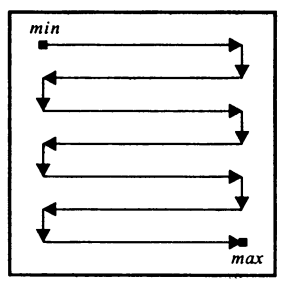
C

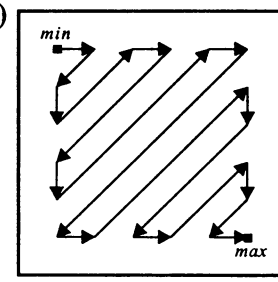
38. Distanța dintre două mulțimi în plan se consideră distanța dintre două puncte ale acestor mulțimi, situate cel mai aproape unul de altul. Aflați distanța dintre două mulțimi de puncte date.

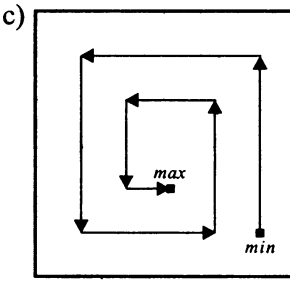
39. În mulțimea dată de puncte în plan aflați două puncte, distanța dintre care este:
a) maximă; b) minimă.

40. Să se afișeze la ecran toate aranjamentele numerelor 1, 2, ..., n, unde n este număr natural nenul dat.

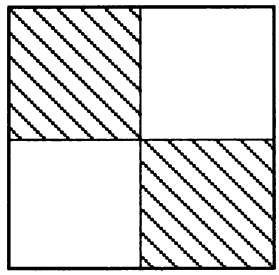
41. Se dă matricea $A(n, n)$ de numere întregi. Să se construiască matricea $B(n, n)$ formată din componentele matricei A conform schemei:

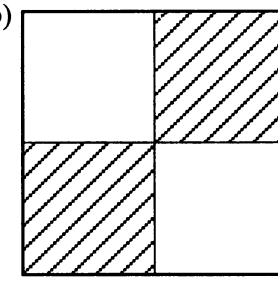
a) 

b) 

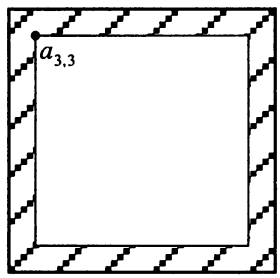
c) 

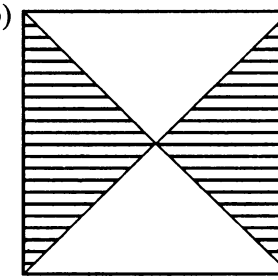
42. Se dă matricea $A(n, n)$ de numere întregi, unde n este număr par. Să se calculeze suma elementelor domeniului haşurat:

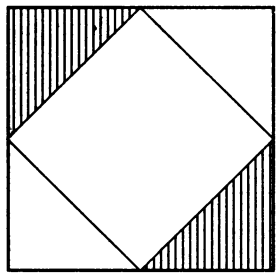
a) 

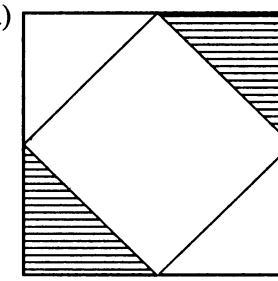
b) 

43. Se dă matricea $A(n, n)$ de numere întregi, unde n este număr impar. Să se calculeze suma elementelor domeniului haşurat:

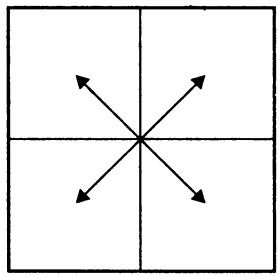
a) 

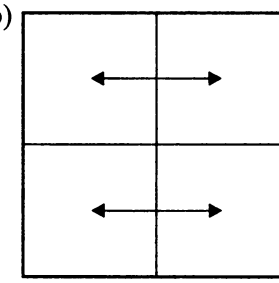
b) 

c) 

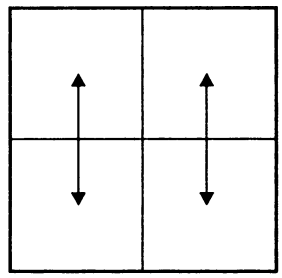
d) 

44. Se dă matricea $A(n, n)$, unde n este număr par. Schimbaţi locurile elementelor ei conform schemei:

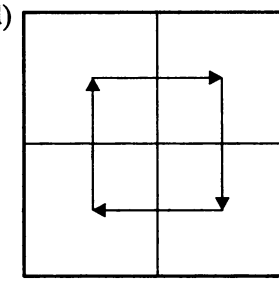
a) 

b) 

c)



d)



45. Se dă matricea $A(m, n)$ ale cărei componente sînt numere întregi. Să se determine toate punctele „şa" şi poziţia lor. Componenta a_{ij} se numeşte punct „şa" dacă ea este componentă minimală în linia i şi componentă maximală în coloana j.

46.* Se dă matricea $A(m, n)$ cu elemente numere întregi. Să se calculeze determinantul matricei.

47.* Se dă matricea $A(m, m)$ cu elemente numere întregi. Să se determine inversa matricei A.

48. Se consideră un grup de persoane în care fiecare două sînt sau nu prieteni (evident că dacă A este prieten cu B, atunci şi B este prieten cu A). Să se găsească perechea de persoane (eventual perechile), care au cei mai mulţi prieteni comuni.

49. Se dă matricea $A(m, n)$ de numere naturale. Să se calculeze cîte elemente „kente" sînt în fiecare linie. Vom numi element „kent" într-un şir de numere elementul care este multiplu comun al succesorului şi predecesorului. Dacă unul din aceste două elemente lipseşte, acesta se consideră egal cu 1.
De exemplu: În şirul 1, 2, 1, 6, 3, 12, 4, 5, 30, 6, 12, 4, 8
elemente „kente" sînt: 2, 6, 12, 30, 12, 8.

50. Se dă matricea $A(m, m)$ cu elemente numere întregi. Să se obţină matricea $B(m, m)$, unde b_{ij} este elementul maximal din dreptunghiul cu vîrfurile în elementele $a_{ii}, a_{jj}, a_{ij}, a_{ji} (i, j = \overline{1,m})$.

51. Fie 4 profesori şi 4 grupe. Să se modeleze orarul a 3 lecţii, astfel încît o grupă să nu aibă 2 lecţii cu acelaşi profesor. Cîte soluţii are problema?
Exemplu: Considerăm matricele $A^k(3, 4)$, unde $k = \overline{1, 4}$, în care $a_{ij}^k = 1$, dacă profesorul j are lecţia i cu grupa k, în caz contrar, $a_{ij}^k = 0$.

Grupa 1

Nr. lecţiei	Profesorul 1	2	3	4
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0

Grupa 2

Nr. lecţiei	Profesorul 1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	0	1

Grupa 3		Profesorul			
		1	2	3	4
Nr. lecției	1	0	0	0	1
	2	0	1	0	0
	3	0	0	1	0

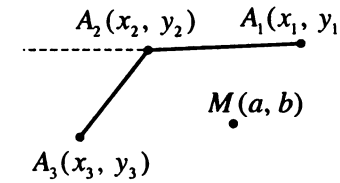
Grupa 4		Profesorul			
		1	2	3	4
Nr. lecției	1	0	0	1	0
	2	0	0	0	1
	3	1	0	0	0

52. Se dau numărul natural $n > 2$, coordonatele vîrfurilor consecutive ale unui poligon convex cu n laturi şi punctul $M(x, y)$. Să se determine poziţia punctului M faţă de poligon (aparţine poligonului, interiorului sau exteriorului lui).

Indicaţie. Fie $Ax + By + C = 0$ ecuaţia dreptei d şi $X(x_0, y_0)$ un punct.

Numărul $Ax_0 + By_0 + C$ se numeşte *abaterea punctului X* de la dreapta d.

Orice dreaptă împarte planul în două semiplane: un semiplan conţine puncte cu abateri pozitive de la dreapta dată, celălalt – cu abateri negative.



În cazul în care punctul dat $M(a, b)$ aparţine interiorului poligonului, pentru oricare 3 vîrfuri $A_1(x_1, y_1)$, $A_2(x_2, y_2)$, $A_3(x_3, y_3)$ consecutive, abaterea punctului M de la dreapta A_1A_2 are acelaşi semn ca şi abaterea punctului A_3 de la dreapta A_1A_2.

Ecuaţia dreptei ce conţine punctele A_1, A_2 se obţine din relaţia $\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$.

53. Se dau numărul natural $n > 2$ şi coordonatele consecutive ale unui poligon. Să se verifice dacă poligonul este convex.

54. *Pătrate magice*

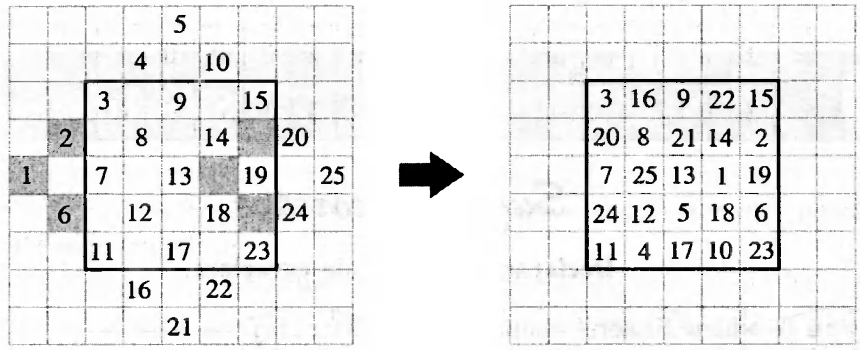
În pătratul alăturat sînt scrise numerele 1, 2, ..., $9 = 3^2$, astfel încît suma numerelor fiecărei linii, a fiecărei coloane şi a fiecărei dintre cele două diagonale este una şi aceeaşi (egală cu 15). Acest pătrat se numeşte pătrat magic de ordinul 3.

2	7	6
9	5	1
4	3	8

Cum se pot obţine astfel de pătrate? În 1612 matematicianul francez Claude Bachet (1581(87) – 1638(48)) a descris următoarea metodă simplă de construire a pătratelor magice de orice ordin impar n:

1. Pe o reţea de pătrate, scriem numerele 1, 2, ..., n^2 pe diagonale, astfel încît să obţinem un pătrat de ordinul n cu laturile pe diagonale.
2. Selectăm în centrul lui un pătrat de ordinul n cu laturile verticale sau orizontale.
3. Fiecare „triunghi de numere din afara" pătratului selectat îl translăm în interiorul pătratului spre latura opusă. Numerele „triunghiului" vor fi plasate în celulele libere.

Obţinem un pătrat magic de ordinul n cu suma numerelor pe fiecare diagonală, linie, coloană egală cu $S_n = \frac{(n^2+1)n}{2}$.



$$S_5 = \frac{5^2+1}{2} \cdot 5 = 65$$

Se dă numărul natural impar n. Să se construiască şi să se afişeze la ecran pătratul magic de ordinul n.

55. *Pătrate latine*

Se dă numărul natural n, $n > 3$. Să se construiască un pătrat latin de ordinul n – un pătrat format din numerele 1, 2, ..., n, astfel încît fiecare număr apare o singură dată în fiecare coloană, linie şi în fiecare dintre cele două diagonale.

De exemplu, pentru $n = 4$ un pătrat latin este:

1	2	3	4
4	3	2	1
2	1	4	3
3	4	1	2

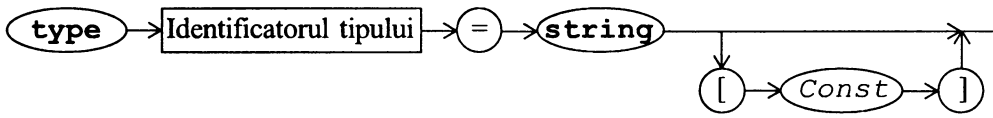
iar pentru $n = 5$ un pătrat latin este:

1	3	5	2	4
5	2	4	1	3
4	1	3	5	2
3	5	2	4	1
2	4	1	3	5

6 Şiruri de caractere

Sugestii teoretice

Declararea unui şir de caractere



`Const` este o constantă de tip **byte** şi indică numărul maximal de simboluri în şir.

Dacă `Const` nu este indicată, atunci implicit lungimea maximală se consideră egală cu 255. Şirul de caractere este similar unui vector ale cărui componente sînt de tip **char** (**array** `[0..const]` **of** `char` sau **array** `[0..255]` **of** `char`).

Pe poziţia 0 se păstrează simbolul al cărui cod ASCII coincide cu lungimea curentă a şirului.

O **constantă-şir** reprezintă o consecutivitate de simboluri inserată între apostrofuri.

Operatori şi funcţii asupra şirurilor de caractere

Operatorul **+** se utilizează pentru **concatenarea** (unirea) cîtorva şiruri.

De exemplu, `'Ana-' + 'Maria'` va returna şirul `'Ana-Maria'`.

Operatorii =, >, <, ≥, ≤, <> se utilizează pentru **compararea şirurilor**. Ei au un ordin de prioritate mai mic decît operatorul +. Se compară simbolurile de pe aceleaşi poziţii, începînd cu poziţia 1, pînă cînd aceste simboluri sînt diferite. Va fi mai mare acel şir, al cărui simbol respectiv este mai mare (amintim că $c_1 < c_2 \Leftrightarrow \text{ord}\ (c_1) < \text{ord}\,(c_2)$).

Dacă două şiruri au lungimi diferite şi unul este subşir al celuilalt, atunci şirul mai lung va fi considerat mai mare.

Exemple: `'Andrei'< 'Vlad'`, deoarece `ord('A')<ord('V')`.

`'Lung'< 'lat'`, deoarece `ord('L')<ord('l')`.

`'stop'< 'stup'`, deoarece `ord('o')<ord('u')`.

`'Egal'< 'Egal '`, deoarece al doilea şir are lungimea mai mare.

Funcţia **Length** (*s*) returnează lungimea (de tip integer) a şirului *s*.

Funcţia **Copy** (*s*,*p*,*n*) returnează din şirul *s* un subşir de lungime *n* începînd cu poziţia *p*.

Funcţia **Concat** ($s_1, s_2, \ldots, s_n$) returnează şirul $s_1+s_2+\ldots+s_n$.

Funcţia **Pos** (*sub*, *s*) returnează 0, dacă şirul *sub* nu este subşir al lui *s*, sau poziţia (de tip integer) din care prima dată subşirul *sub* apare în *s*.

Procedura **Delete** (*s*, *p*, *n*) şterge *n* simboluri din şirul *s* începînd cu poziţia *p*.

Procedura **Insert** (*sub*, *s*, *p*) inserează subşirul *sub* în şirul *s* începînd cu poziţia *p*.

Procedura **Str** (*x*, *s*) transformă numărul *x* (de tip integer sau real) în şir, atribuindu-l lui *s*. Parametrul *x* poate avea una din următoarele forme de reprezentare:

a) X

b) $X : m$

c) $X : m : f$,

unde X este expresia (sau parametrul) a cărei valoare va fi transformată, m reprezintă numărul minim de caractere ale şirului, iar f (se scrie doar cînd X este real) – numărul de simboluri ale părţii fracţionare.

Procedura **Val** (*s*, *x*, *cod*) transformă şirul *s* în număr, atribuindu-l lui *x* (de tip întreg sau real). Valoarea lui *cod* devine 0, dacă transformarea a fost reuşită, altfel *cod* conţine poziţia unde a fost întîlnit un simbol nepermis. Şirul *s* poate conţine spaţii de debut.

Exemple:

Valoarea lui *s* (sau a lui *x*)	Expresia	Rezultatul funcţiei sau valoarea finală a parametrilor (în cazul procedurii)
'Informatica'	Length(s)	11
'A sosit iarna!'	Length(s)	14
'Informatica'	Copy(s,3,5)	'forma'
'Tractor'	Copy(s,3,5)	'actor'
'Tractor'	Concat(s,'ist')	'Tractorist'
'Informatica'	Pos('forma',s)	3
'Informatica'	Pos('Forma',s)	0
'Tractor'	Delete(s,1,2)	s←'actor'
'Informatica'	Delete(s,3,8)	s←'Ina'
'bine'	Insert('ul',s,2)	s←'buline'
x←28	Str(x,s)	s←'28'
s←'341'	Val(s,x,cod)	x←341 cod←0
s←'3,14' x←10	Val(s,x,cod)	x←10 cod←2

Probleme rezolvate

❶ Se dă un text (şir de caractere). Să se afişeze literele de pe poziţiile pare.

Rezolvare:

```
program Sir1;
uses Crt;
var s:string;
    i:byte;
BEGIN
  ClrScr;
```

```
   write ('Scrie textul: ');
   readln(s);
   i:=2;
   while i<=length(s) do begin
      write(s[i],' ');
      i:=i+2;
   end;
   readkey;
END.
```

❷ Se dă un text. Se se calculeze numărul literelor 'm' (mari şi mici) din acest text.

Rezolvare:

```
program Sir2;
uses Crt;
var s:string;
    i,n:byte;
BEGIN
   ClrScr;
   write('Scrie textul: ');
   readln(s);
   n:=0;
   for i:=1 to length(s) do
      if UpCase(s[i])='M' then inc(n);
   write('n= ',n);
   readkey;
END.
```

❸ Un cuvînt se numeşte *palindrom* dacă el coincide cu „răsturnatul" său. De exemplu, cuvintele „potop", „cazac", „cojoc" sînt palindroame.

Să se scrie un program care va verifica dacă este cuvîntul dat palindrom.

Rezolvare:

```
program Sir3;
uses Crt;
var s,r:strinng; {r este rasturnatul lui s}
    i:byte;
BEGIN
   ClrScr;
   write('Scrie textul: ');
   readln(s);
   for i:=1 to length(s) do
      r:=s[i]+r;
   if s=r then write('Este palindrom') else
      write('Nu este palindrom');
   readkey;
END.
```

Observaţie

„Răsturnatul" lui *s* putea fi construit şi astfel:

```
      for i=length(s) downto 1 do
         r:=r+s[i];
```

Observaţie

Problema putea fi rezolvată şi fără a construi „răsturnatul" lui *s*. Fie *n* lungimea lui *s*. Atunci *s* este palindrom dacă şi numai dacă:

s[1] = s[n], s[2] = s[n–1], ..., s[trunc (n/2)] = s[n – trunc (n/2) +1], adică

s[i] = s[n – i + 1], $\forall i = \overline{1, \text{trunc}(n/2)}$.

Astfel, secvenţa de verificare se va scrie:

```
f:=true; {f este de tip boolean}
n:=length(s);
for i:=1 to trunc(n/2) do
   if s[i]<>s[n-i+1] then f:=false;
if f then write('Este palindrom') else
   write('Nu este palindrom');
```

❹ Să se determine dacă textul dat conţine două simboluri alăturate identice.

```
program Sir4;
uses Crt;
var s, m: string;
    i: byte;
BEGIN
   ClrScr;
   write('Scrie textul: ');
   readln(s);
   m:='Nu contine';
   for i:=1 to length(s)-1 do
      if s[i]=s[i+1] then m:='Contine';
   write(m);
   readkey;
END.
```

❺ Se dau numerele naturale *a* şi *b* mai mici decît 30000 şi o cifră. Să se verifice dacă cifra dată se conţine în suma *a* + *b*.

Rezolvare:

```
program Sir5;
uses Crt;
var a,b,x:word; {x este suma}
    s,m:string;
    c:char; {cifra}
    i:byte;
BEGIN
     ClrScr;
     write('Scrie numerele: ');
     readln(a,b);
     write('Scrie cifra: ');
     readln(c);
     x:=a+b;
     str(x,s); {transformam suma x in textul s}
     m:='Nu se contine';
     for i:=1 to length(s) do
        if s[i]=c then m:='Se contine';
```

```
    {Se putea si astfel if pos(c,s)<>0 then m:='Se contine'}
    write(m);
    readkey;
END.
```

❻ Se dă un text în care cuvintele sînt separate prin unul sau mai multe spaţii. Să se calculeze cîte cuvinte conţine textul.

Rezolvare:

Dacă textul are forma:

cuvîntul 1 cuvîntul 2 ... cuvîntul n

atunci trebuie doar să numărăm spaţiile şi să adăugăm o unitate la rezultat.

De aceea, pentru orice eventualitate:

- vom lichida spaţiile de debut;
- vom lăsa între cuvinte doar un spaţiu.

```
program Sir6;
uses Crt;
var s:string;
    i,k:byte;
BEGIN
   ClrScr;
   write('Scrie textul: ');
   readln(s);
   {lichidam spatiile de debut}
   while s[1]=' ' do delete(s,1,1);
   {lasam intre cuvinte doar un spatiu}
   i:=1;
   repeat
      if copy(s,i,2)='  ' then delete(s,i,1) else inc(i);
   until i>length(s);
   k:=1;
   for i:=1 to length(s) do
      if s[i]=' ' then inc(k);
   write('Numar de cuvinte: ',k);
   readkey;
END.
```

Observaţie

Programul nu va afişa nimic dacă textul va consta doar din spaţii. Lichidaţi acest neajuns.

❼ Se citesc de la tastatură 2 string-uri (texte) formate din cifre, fiecare reprezentînd un număr natural. Să se calculeze suma acestor numere.

Rezolvare:

Fie *s*1 şi *s*2 şirurile date. Considerăm lungimea şirului *s*1 mai mare sau egală cu lungimea şirului *s*2. Vom aplica algoritmul adunării în coloană. Deci, începînd de la ultima cifră, adunăm unităţile. Cifra unităţilor rezultatului va fi restul împărţirii la 10 a sumei unităţilor numerelor date. Cîtul împărţirii la 10 a sumei unităţilor numerelor date va fi numărul plasat

în memorie. Cifra zecilor rezultatului va fi restul împărțirii la 10 a sumei zecilor numerelor date plus numărul din memorie ş.a.m.d.

```
program Sir7;
uses crt;
var s1,s2,s3,s:string;
    i,j:byte;
    x,y,m,code:integer; {m - memoria}
BEGIN
  Clrscr;
  write('Introdu primul numar: ');
  readln(s1);
  write('Introdu numarul al 2-lea: ');
  readln(s2);
  if length(s2)>length(s1) then begin
     s3:=s1;
     s1:=s2;
     s2:=s3;
  end;
  s3:='';
  j:=length(s2);
  m:=0;
  s:='';
  for i:=length(s1) downto 1 do begin
     val(s1[i],x,code);
     val(s2[j],y,code);
     if j>0 then str((x+y+m) mod 10,s3);
     if j<=0 then str((x+m) mod 10,s3);
     m:=(x+y+m) div 10;
     j:=j-1;
     s:=s3+s;
     if (i=1) and (m<>0) then begin
        str(m,s3);
        s:=s3+s;
     end;
  end;
  writeln('Suma este: ',s);
  readkey;
END.
```

⑧ De la tastatură se citeşte un text. Textul apare în rîndul 15, apoi se mişcă pe orizontală pînă dispare, ca să apară din nou începînd cu prima coloană.

Rezolvare:

```
program Sir8;
uses Crt;
var s:string;
    i:integer;
BEGIN
  ClrScr;
  write('Scrie textul: ');
  readln(s);
  for i:=1 to 80 do
  s:=' '+s+' ';
```

```
    {adaugam cite 80 de spatii la inceputul si sfirsitul textului}
    ClrScr;
    i:=1;
    while not KeyPressed do begin
       gotoxy(i,15);
       write(s);
       delay (200);
       inc(i);
       if i=80 then i:=1;
    end;
END.
```

Probleme propuse

A

1. Să se scrie un program care întreabă numele utilizatorului, apoi îl salută.
2. Se dă un text. Să se transforme toate literele în majuscule.
3. Se dă un text. Să se afişeze literele de pe poziţiile:
   a) pare; b) impare.
4. Se dă un text. Să se calculeze numărul literelor „a" din text.
5. Se dă un text. Să se calculeze numărul vocalelor din text.
6. Se dă un text. Să se afişeze „răsturnatul" textului dat. De exemplu, pentru textul „tractor" se va afişa „rotcart".
7. Se dă un text. Să se verifice dacă textul este palindrom (egal cu „răsturnatul" său).
8. Se dă un text. Să se calculeze de cîte ori se întîlneşte în acest text silaba „oa".
9. Se dă un text. Să se substituie în acest text literele „o" (mari şi mici) prin litera „u" (respectiv, mare sau mică). De exemplu, pentru textul „Om frumos" se va afişa „Um frumus".
10. Se dă un text. Să se substituie literele „a" cu litera „o" şi invers. De exemplu, pentru textul „program" se va afişa „pragrom".
11. Se dă un text. Să se substituie în acest text litera „a" prin silaba „oa". De exemplu, pentru textul „capac" se va afişa „coapoac".
12. Se dă un text. Să se substituie litera „a" prin combinaţia „cu" şi invers. De exemplu, pentru textul „culoare" se va afişa „alocure".
13. Se dă un text. Să se substituie litera „a" prin „o", dacă litera „a" se află pe poziţie impară, şi prin „e", dacă litera „a" se află pe poziţie pară.
    De exemplu, pentru textul „calcar" se va afişa „celcor".

14. Se dă un text. Să se substituie în acest text combinația „cs" prin litera „x".

15. Se dă un cuvînt. Să se delimiteze în acest cuvînt literele „a" prin spații. De exemplu, pentru cuvîntul „tractor" se va afișa „tr a ctor".

16. Se dă un text. Să se dubleze fiecare literă din text. De exemplu, pentru textul ' tort" se va afișa textul „ttoorrtt".

17. Se dă un text. Să se determine care dintre literele „a", „o" se întîlnește prima în text.

18. Se dă un text și o literă. Să se determine dacă litera dată se conține în textul dat. În caz afirmativ, se afișează prima poziție a literei, altfel – mesajul corespunzător.

19. Se dau textele x și y. Să se determine dacă textul y se conține în x. În caz afirmativ, se afișează prima poziție a lui y în x, altfel – mesajul corespunzător.

20. Se dă un text. Să se elimine combinația „ea" din acest text. De exemplu, pentru textul „Luceafarul este o stea" se va afișa „Lucfarul este o st".

21. Se dă un text. Să se insereze litera „o" înaintea fiecărei litere „a" precedate de litera „n". De exemplu, pentru textul „canal" se va afișa „canoal".

22. Se dă un text. Să se deplaseze spre dreapta cu o poziție fiecare literă, astfel încît litera de pe poziția n să fie plasată pe poziția $n + 1$, iar ultima literă să devină prima. De exemplu, pentru textul „tractor" se va afișa textul „rtracto".

23. Se dă un text. Să se deplaseze spre stînga cu o poziție fiecare literă, astfel încît litera de pe poziția n să fie plasată pe poziția $n - 1$, iar prima literă să devină ultima. De exemplu, pentru textul „tractor" se va afișa textul „ractort".

24. Se dă un text. Să se afișeze litere alfabetului latin care nu apar în acest text.

25. Se dă un cuvînt. Să se substituie fiecare literă cu un spațiu precedat de numărul de ordine al acestei litere în alfabetul latin. De exemplu, pentru cuvîntul „lac" se va afișa „12 1 3 ".

26. Se dă un text.
a) Să se codifice acest text, substituind fiecare simbol c cu un simbol, al cărui cod (în tabelul ASCII) este cu 2 unități mai mare decît codul lui c.
b) Să se scrie alt algoritm care va decodifica textul obținut în a).

27. Se dă un text.
a) Să se codifice acest text, schimbînd locurile simbolurilor de pe pozițiile 1 și 3, 2 și 4, 3 (noul 3) și 5 ș.a.m.d. De exemplu, pentru textul inițial „proba" se va obține textul „obarp".
b) Să se scrie alt algoritm care va decodifica textul obținut în a).

28. Se dă un text.

a) Să se codifice acest text, schimbînd locurile simbolurilor de pe pozițiile 1 și 2, 3 și 4 ș.a.m.d., apoi să se substituie fiecare simbol *c* cu un simbol, al cărui cod (în tabelul ASCII) este cu 2 unități mai mare decît codul lui *c*. De exemplu, pentru textul inițial „abac" se va obține textul „dcec" („abac" → „baca" → „dcec").

b) Să se scrie alt algoritm care va decodifica textul obținut în a).

29. Se dă un text.

a) Să se codifice acest text, substituind fiecare simbol *c* cu un simbol, al cărui cod (în tabelul ASCII) este cu *r*(*i*) unități mai mare decît codul lui *c*, unde *r*(*i*) este restul împărțirii la 3 a poziției *i* a lui *c* în text.

b) Să se scrie alt algoritm care va decodifica textul obținut în a).

30. Se dă un text. Să se suprime din text toate spațiile în plus, astfel încît între fiecare două cuvinte să rămînă doar un spațiu.

31. Se dă un text care conține paranteze. Să se suprime textul din paranteze împreună cu parantezele. (Se consideră că între paranteze nu există alte paranteze.)

32. Se dă un text. Să se transforme toate literele în litere mici. De exemplu, pentru textul „Ion a plecat la Tiraspol" se va afișa „ion a plecat la tiraspol".

33. Se dă un text. Să se transforme literele mari în litere mici și invers. De exemplu, pentru textul „A sosit vara" se va afișa „a SOSIT VARA".

B

34. Se dau textele *x* și *y*. Să se verifice dacă, permutînd literele textului *x*, putem obține textul *y*. De exemplu, pentru *x* = „cal" și *y* = „lac" se va afișa „da", iar pentru *x* = „copac" și *y* = „capac" se va afișa „nu".

35. Se dă un cuvînt. Să se determine numărul de litere diferite ale acestui cuvînt.

36. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze fiecare cuvînt din rînd nou.

37. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze cuvintele care conțin exact 2 litere „a".

38. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze cuvîntul de lungime maximală. Dacă astfel de cuvinte sînt mai multe, să se afișeze toate.

39. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze cuvintele textului în ordine alfabetică (lexicografică).

40. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Pentru fiecare cuvînt din text să se insereze ultima literă a cuvîntului înaintea primei.
De exemplu, pentru textul „A fost odata " se va afișa „AA tfost aodata".

41. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Pentru fiecare cuvînt din text să se adauge prima literă a cuvîntului după ultima.
De exemplu, pentru textul „O zi cu soare" se va afișa „OO ziz cuc soares".

42. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se verifice dacă în text există cuvinte care se repetă.

43. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze cuvîntul care se repetă de cele mai multe ori în text. Dacă astfel de cuvinte sînt mai multe, să se afișeze toate.

44. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se substituie fiecare cuvînt cu „răsturnatul" său.
De exemplu, pentru textul „Culegere de probleme" se va afișa „eregeluC ed emelborp".

45. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze textul inversînd ordinea cuvintelor.
De exemplu, pentru textul „Culegere de probleme la informatica" se va afișa „informatica la probleme de Culegere".

46. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se afișeze cuvintele diferite.

47. Se dă un text în care cuvintele sînt separate prin unul sau mai multe spații. Să se suprime din text cuvintele de pe poziții pare.
De exemplu, pentru textul „Culegere de probleme pentru concursuri" se va afișa „Culegere probleme concursuri".

48. Unele numere naturale reprezintă ultimele cifre ale pătratelor lor. (De exemplu, $5 \cdot 5 = 25$, $25 \cdot 25 = 625$). Să se afișeze la ecran toate numerele naturale cu proprietatea menționată, mai mici decît 60 000.

49. *Ne amuzăm și rezolvăm*
Radu a compus o poezie despre fata visurilor lui – Doina. Între timp visul s-a realizat, însă cu o altă fată – Mirela. S-a gîndit Radu că ar fi păcat să nu păstreze poezia, dar trebuie să substituie în ea toate cuvintele „Doina" prin „Mirela". Ajutați-l pe Radu, realizînd un algoritm de substituție.

50. Se dă un număr natural, scris cu cifre arabe (0, 1, ..., 9). Să se scrie cu cifre romane (I, V, X, L, C, D, M) numărul dat. De exemplu, pentru 1971 se va afişa MCMLXXI (1000 + (1000 – 100) + (50 + 10 + 10) + I).

51. Se dă un număr natural scris cu cifre romane (I, V, X, L, C, D, M). Să se scrie cu cifre arabe (0, 1, ..., 9) numărul dat.
De exemplu, pentru MDCIX se va afişa 1609 (1000 + 500 + 100 + (10 – 1)).

7 Numere aleatoare

Sugestii teoretice

Procedura **Randomize** conectează generatorul de numere aleatoare.

Funcția **Random**[(*n*)], unde n este de tip word, returnează un număr natural aleator din intervalul [0,*n*-1]. Dacă parametrul n lipsește, atunci funcția returnează un număr real din intervalul [0,1).

Astfel, pentru a genera un număr întreg din intervalul [*a*,*b*] se va scrie expresia *random*(*b*-*a*+1)+*a*, iar pentru a genera un număr real din intervalul [*a*,*b*) se va scrie expresia *random**(*b*-*a*)+*a*.

Exemple rezolvate

❶ Să se genereze un număr întreg cu modulul mai mic decît 10.

Rezolvare:

Fie a numărul ce urmează a fi generat. Atunci $|a|<10$, $a\in\mathbb{Z}$, deci $a\in[-9, 9]$.

```
program Aleat1;
uses Crt;
var a:integer;
BEGIN
   ClrScr;
   Randomize;
   a:=random(19)-9; {random (9-(-9)+1)+9}
   writeln('Numar aleator intreg cu modulul mai mic decit 10: ',a);
   readkey;
END.
```

❷ Să se genereze un număr real cu modulul mai mare decît modulul numărului real dat R, $R<10000$, și mai mic decît 10000.

Rezolvare:

Fie X numărul ce urmează a fi generat.

Deci, X aparține intervalului $(-10000,-|R|)\cup(|R|,10000)$. Prin urmare, vom genera modulul numărului X (adică un număr natural de pe intervalul $(|R|,10000)$), apoi semnul lui.

```
program Aleat2;
uses Crt;
var R,X:real;
    s:integer;
BEGIN
   ClrScr;
```

```
   write('Introdu R: ');
   readln(R);
   Randomize;
   X:=random*(10000-abs(R))+abs(R);{generam modulul lui X}
   s:=random(2);{generam unul din numerele 0 (corespunzator semnului + )
      sau 1 (corespunzator semnului -)};
   if s=1 then X:=-X;
   write('|',R:3:2,'| < |',X:3:2,'| < 10000');
   readkey;
END.
```

❸ Să se genereze 10 numere naturale mai mici decît 50, diferite fiecare două.

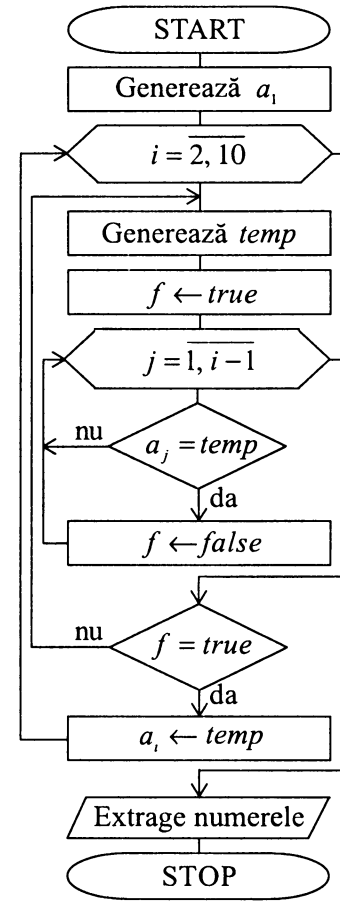
Rezolvare:

Vom păstra numerele în vectorul $A(10)$.

Fie că dorim să generăm componenta i.

Inițial atribuim valoarea generată variabilei *temp*.

Dacă valoarea *temp* nu coincide cu nici una dintre valorile primelor $i-1$ componente, atunci componentei a_i i se atribuie valoarea variabilei *temp*, altfel generăm din nou valoarea variabilei *temp*.




```
program Aleat3;
uses Crt;
var a:array[1..10] of byte;
    i,j,temp:byte;
    f:boolean;
BEGIN
   ClrScr;
   Randomize;
   a[1]:=Random(50);
   for i:=2 to 10 do begin
      repeat
         temp:=random(50);
         f:=true;
         for j:=1 to i-1 do
            if a[j]=temp then f:=false;
      until f;
      a[i]:=temp;
   end;
   {Afişam numerele}
   for i:=1 to 10 do
      write(a[i],' ');
   readkey;
END.
```

❹ Să se alcătuiască un joc-test de verificare a tablei înmulțirii de la 0 la 10. Testul propune aleator 20 de înmulțiri, după care afișează nota, calculată după formula `round (Rc/2)`, unde `Rc` este numărul de răspunsuri corecte. De exemplu, pentru 10 răspunsuri corecte se va afișa nota 5, iar pentru 11 – nota 6.

Rezolvare:

```
program Test;
uses Crt;
var p,f1,f2,nota,rc,i:integer;
BEGIN
   ClrScr;
   writeln('TEST CU TABLA INMULTIRII');
   Randomize;
   rc:=0;
   for i:=1 to 20 do begin
     f1:=random(11);
     f2:=random(11);
     write(f1,' X ',f2,' = ');
     readln(p);
     if p=f1*f2 then inc(rc);
   end;
   nota:=round(rc/2);
   writeln('Ai obtinut nota ',nota);
   readkey;
END.
```

Probleme propuse

A

1. Să se genereze un număr natural mai mic decît numărul natural dat n.
2. Să se genereze un număr natural mai mare decît numărul natural dat n şi mai mic decît numărul natural dat m, unde $n < m$.
3. Să se genereze un număr real pozitiv mai mic decît 1.
4. Să se genereze un număr real negativ mai mare decît –0,5.
5. Să se genereze 3 numere reale negative distincte mai mari decît –2.
6. Să se genereze 3 numere reale distincte mai mari decît 2 şi mai mici decît 4.
7. Să se genereze 5 numere reale distincte mai mari decît –3 şi mai mici decît 2.
8. Să se genereze o literă mică a alfabetului latin.
9. Să se genereze o literă mare a alfabetului latin.
10. Să se genereze un număr natural de 3 cifre divizibil cu 6 şi cu 8.
11. Să se genereze un număr natural prim mai mare decît 100.
12. Să se genereze un vector ale cărui n ($1 < n < 50$) componente sînt numere naturale, distincte fiecare două, mai mici decît 100.

13. Să se genereze un tablou bidimensional $A(m, n)$, unde $1 < m < 15$ şi $1 < n < 15$, ale cărui componente sînt numere naturale, diferite fiecare două, mai mici decît 300.

14. Să se genereze 10 variante „Superloto 5 din 36".

15. *Rezolvăm, apoi ne jucăm*
Să se creeze un joc, în care calculatorul propune jucătorului să ghicească din 10 încercări un număr natural aleator mai mic decît 1 000. În urma fiecărei încercări se va afişa un mesaj care va indica dacă numărul propus de jucător este mai mare sau mai mic decît cel căutat.
Jocul va conţine mesaje de felicitare, de regret etc.

16. *Rezolvăm, apoi ne jucăm*
Să se creeze un joc, în care calculatorul propune jucătorului să ghicească din 6 încercări o literă generată. (Nu va conta dacă litera este mare sau mică.) În urma fiecărei încercări se va afişa un mesaj care va indica dacă litera propusă de jucător urmează sau precede în ordinea alfabetică litera căutată.
Jocul va conţine mesaje de felicitare, de regret etc.

17. Să se modeleze „cerul înstelat" (în regim textual.)
Indicaţie. Se va utiliza procedura `GotoXY`.

18. Să se genereze cuvîntul „program". Pentru fiecare literă să se afişeze numărul de apelări la generator.

19. Pentru fiecare literă a alfabetului latin, să se genereze aleator un număr natural mai mic decît 27, astfel încît oricăror 2 litere să li se asocieze 2 numere diferite.

20. Să se genereze 3 litere mici diferite ale alfabetului latin.

21. Să se genereze 5 litere mari diferite ale alfabetului latin.

22. Să se genezeze un număr întreg cu modulul mai mic decît 1 000 şi mai mare decît 20.

B

23. Să se genereze 10 litere diferite ale alfabetului latin, printre care 3 mici şi 7 mari.

24. Să se genereze 10 litere diferite ale alfabetului latin, printre care cel puţin 3 mici.

25. Să se genereze 10 litere diferite ale alfabetului latin, printre care cel mult 3 mari.

26. Să se genereze o vocală mică a alfabetului latin.

27. Să se genereze o vocală mare a alfabetului latin.

28. Să se genereze o consoană mică a alfabetului latin.

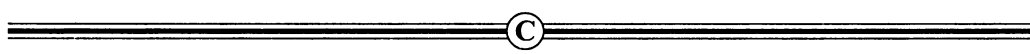
29. Să se genereze o consoană mare a alfabetului latin.

30. Să se genereze 5 numere prime diferite mai mici decît 1 000.

31. Să se genereze de 5 000 de ori cîte un număr întreg cu modulul mai mic decît 10.
a) Să se calculeze raportul dintre numărul numerelor negative apărute şi cel al numerelor pozitive. Rezultatul să se compare cu numărul 0,5.
b) Să se calculeze numărul de apariţii ale numărului: 1, 2, –3, 5, –6. Ce se observă?

32. Să se alcătuiască un joc-test de verificare:
a) a adunării numerelor de la 0 la 100;
b) de scădere a numerelor de la 0 la 100;
c) de împărţire a numerelor de la 0 la 100.

33. Să se deseneze pe ecran pătrate de dimensiuni aleatoare, în poziţii aleatoare şi de culori aleatoare, pînă se acţionează o tastă.
Indicaţie. Să se utilizeze procedura `TextBackGround(C:byte)`, care stabileşte culoarea `C` pentru fundalul ecranului.



34. *Rezolvăm, apoi ne jucăm*
Să se modeleze jocul „Cîmpul minunilor" pentru 5 teme şi 15 cuvinte pentru fiecare temă.

35. *Rezolvăm, apoi ne jucăm*
Să se modeleze jocul „Vrei să fii miliardar?" pentru 5 teme şi 15 întrebări pentru fiecare temă.

36. *Rezolvăm, apoi ne jucăm*
Să se modeleze un joc în regim textual, care propune jucătorului să ghicească din 3 cărţi de joc (2 „valeţi" şi un „as") „asul".
Faţa cărţilor poate arăta astfel:

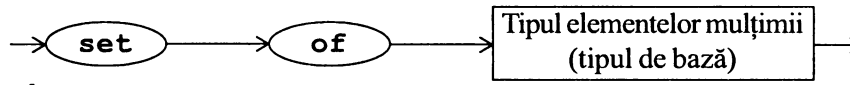
```
**********          **********
*  *******  *        ******  ******
**  *****  **        *****    *****
***  ***  ***        ****  *  ****
****  *  ****        ***      ***
*****    *****       **  *****  **
******  ******       *  *******  *
**********          **********
```

Jocul va conţine mesaje de felicitare, de regret etc.

8 Tipul de date *mulțime* (Set)

Sugestii teoretice

Declararea tipului de date *mulțime*



Exemple:

```
var cifre: set of '0'..'9';
    a: set of byte;
    b: set of 0..100;
    decor: set of (rosu, galben, albastru, verde, oranj, alb);
```

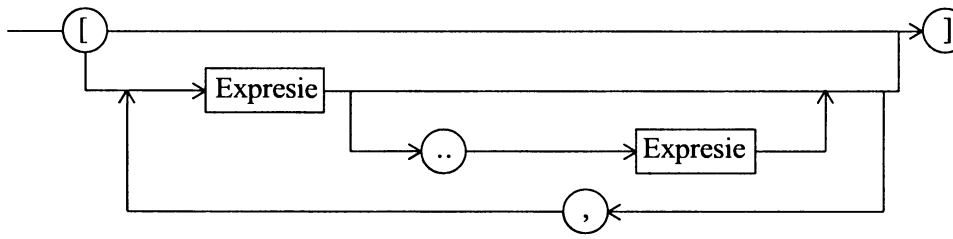
Observații

1. Tipul elementelor mulțimii trebuie să fie ordinal și numărul lor total nu trebuie să fie mai mare decît 256.

2. Numărul de ordine al limitei inferioare și al celei superioare ale tipului de bază trebuie să aparțină intervalului [0, 255].

3. Dacă tipul de bază are *n* valori, atunci tipul de date *mulțime* va avea 2^n valori.

Forma constructorului de mulțime



Exemple:

```
a:=[1, 2, 5];
b:=[x..x + 9];
cifre:=['9'];
decor:=[verde, pred(galben)]
```

Observație

Constructorul [ ] reprezintă mulțimea vidă.

Operatori aplicabili asupra mulțimilor

+	reuniunea	$A + B$ returnează $A \cup B$
*	intersecția	$A * B$ returnează $A \cap B$
–	diferența	$A - B$ returnează $A \setminus B$
=	egalitatea mulțimilor	$A = B$ returnează *true*, dacă $A = B$, altfel – *false*
<>	neegalitatea mulțimilor	$A <> B$ returnează *true*, dacă $A \neq B$, altfel – *false*
<=	incluziunea	$A <= B$ returnează *true*, dacă A se include în B, altfel – *false*
>=		$A >= B$ returnează *true*, dacă B se include în A, altfel – *false*
in	apartenența	a in A returnează *true*, dacă $a \in A$, altfel – *false*

Probleme rezolvate

❶ Fie mulțimile $A = \{1, 3, a, 4, c, d, 5, 8, 2\}$, $B = \{2, a, c, 8, 4, 9, e, 3\}$. Să se calculeze mulțimea $C = (A \cup B) \setminus (A \cap B)$.

Rezolvare:

```
program Set1;
uses Crt;
var a,b,c: set of char;
    i:integer;
BEGIN
   ClrScr;
   A:=['1','3','a','4','c','d','5','8','2'];
   B:=['2','a','c','8','4','9','e','3'];
   C:=(A+B)-(A*B);
   {Afisarea multimii C}
   for i:=1 to 255 do
      if chr(i) in C then write(chr(i),' ');
   readkey;
END.
```

❷ Se dă un text cu cel mult 255 de caractere. Să se afişeze caracterele diferite din text. De exemplu, pentru textul „AbraCadabra" se va afişa „ACabdr"

Rezolvare:

```
program Set2;
uses Crt;
var diferit: set of char;
    text:string;
    i:byte;
BEGIN
   ClrScr;
   write('Scrie textul: ');
   readln(text);
```

```
   diferit:=[];
   for i:=1 to length(text) do
      diferit:=diferit+[text[i]];
   for i:=1 to 255 do
      if chr(i) in diferit then write(chr(i));
   readkey;
END.
```

❸ Se dă numărul natural n, $n < 20$. Se citesc de la tastatură n mulţimi de numere naturale mai mici decît 100. Să se afişeze reuniunea şi intersecţia acestor mulţimi.

Rezolvare:

Vom păstra mulţimile într-un vector. Iniţial vom atribui mulţimii-intersecţie toate numerele de la 0 la 100.

```
program Set3;
uses Crt;
type multime=set of byte;
var a:array[1 ..20] of multime;
    reun,inter:multime;
    i,n,el:byte;
BEGIN
   ClrScr;
   write('Numarul de multimi: ');
   readln(n);
   for i:=1 to n do begin
      a[i]:=[];
      {Scriem elementele multimii i}
      writeln('Multimea ',i);
      repeat
         write('Elementul: ');
         readln(el);
         a[i]:=a[i]+[el];
      until not (el in [0..100]);
   end;
   reun:=[];
   inter:=[0..100];
   for i:=1 to n do begin
      reun:=reun+a[i];
      inter:=inter*a[i];
   end;
   writeln('Reuniunea');
   for i:=0 to 100 do
      if i in reun then write(i,' ');
   writeln;
   writeln('Intersectia');
   for i:=0 to 100 do
      if i in inter then write(i,' ');
   readkey;
END.
```

❹ Se consideră un grup din n persoane. Pentru orice pereche (i, j) din grup, se știe dacă persoana i, aflînd o noutate, o va transmite persoanei j sau nu. Se dă numărul natural p, $p \le n$, considerînd că persoana p a aflat un secret. Să se stabilească dacă toți membrii grupului vor afla secretul.

Rezolvare:

Vom construi matricea $A(n, n)$ în care $a_{ij} = \begin{cases} 1, \text{ dacă } i \text{ transmite noutatea lui } j \\ 0, \text{ în caz contrar.} \end{cases}$

Fie X mulțimea persoanelor neinformate. Evident, inițial $X = \{ 1, 2, ..., n\} \setminus \{p\}$. Dacă persoana j va afla noutatea, atunci X devine $X \setminus \{j\}$. Vom citi perechile prin care circulă secretul pînă cînd se va introduce o pereche incorectă.

```
program Set4;
uses Crt;
var  A:array[1..30,1..30] of 0..1;
     n,i,j,p,m:integer;{m va fi mesagerul}
     X: set of 1..30;
BEGIN
   ClrScr;
   write('Introdu numarul de persoane din grup: ');
   readln(n);
   for i:=1 to n do
      for j:=1 to n do
         a[i,j]:=0;
   writeln('Introdu perechile prin care circula noutatile: ');
   readln(i,j);
   while (i in [1..n]) and (j in [1..n]) do begin
      a[i,j]:=1;
      readln(i, j);
   end;
   write('Introdu persoana care prima a aflat secretul: ');
   readln(p);
   X:=[1..n]-[p];
   m:=0;
   repeat
      for i:=1 to n do
         for j:=1 to n do
            if not (i in X) and (j in X) and (a[i,j]=1) then begin
               writeln(i, '->',j);
               X:=X-[j];
            end;
      inc(m);
   until (m>n) or (X=[]);
   if X=[] then write('Da') else write('Nu');
   readkey;
END.
```

Exerciții și probleme propuse

A

1. Care va vi valoarea expresiei:

a) `[0,1,2,3]=[0..3];`
b) `['a','a','a']<>['a'];`
c) `[1,2]=[2,1];`
d) `[3,4,6,8]<=[1..8];`
e) `[2,4..7]<=[1..2,4..8];`
f) `[]-['a','c'..'e'];`
g) `[1..10]>=[];`
h) `'A' in (['A'..'Z']*['a'..'z'];`
i) `round(2.5) in [2, 3, 4]`?

2. Fie declarațiile:

```
var x: set of 0..15;
    y,z:byte;
```

Considerăm `y=4`, `z=3`. Ce valoare va avea `x` în urma execuției instrucțiunii:

a) `x:=[y+z,1..4,sqr(z)];`
b) `x:=[trunc(sqr(y)/z)..10];`
c) `x:=[0..3*z,2..sqr(y)-5]`?

3. Se dă un text. Să se afișeze:

a) vocalele care nu apar în text;
b) consoanele care nu apar în text;
c) cifrele care nu apar în text.

4. Se dă un vector cu n ($1 \le n \le 100$) componente numere naturale mai mici decît 150. Să se afișeze numerele naturale mai mici decît 150, care nu sînt componente ale vectorului.

5. Se dă un text. Să se calculeze numărul:

a) vocalelor;
b) consoanelor;
c) simbolurilor care nu sînt litere;
d) literelor mici;
e) literelor mari;
f) cifrelor.

6. Se dă numărul natural n, $n < 20$. Se citesc de la tastatură n șiruri de caractere (formate din cifre și litere). Să se afișeze caracterele folosite în toate șirurile.

7. Se dau mulțimile X și Y de numere naturale mai mici decît 200. Să se determine mulțimile:

a) $X \cup Y$;
b) $X \cap Y$;
c) $X \setminus Y$;
d) $(X \setminus Y) \cup (Y \setminus X)$;
e) $(X \setminus Y) \cap (Y \setminus X)$.

8. Considerînd X, Y, Z mulțimi de numere naturale date mai mici decît 200 să se verifice legile lui de Morgan[1]:

a) $\overline{X \cup Y \cup Z} = \overline{X} \cap \overline{Y} \cap \overline{Z}$;
b) $\overline{X \cap Y \cap Z} = \overline{X} \cup \overline{Y} \cup \overline{Z}$.

9. Se dă un vector cu n ($1 \le n \le 100$) de componente întregi cu modulul mai mic decît 1000. Să se afișeze în ordine descrescătoare componentele diferite din vector. De exemplu, pentru vectorul –4, –4, 3, 7, 0, –9, 11, 5, 3, 3, 7, 1, –9 se va afișa 11, 7, 5, 3, 1, 0, –4, –9.

---

[1] Augustus de Morgan (1806–1871) – matematician și logician englez.

B

10. Se dă un număr natural n cu cel mult 255 de cifre. Să se afişeze la ecran cel mai mic număr ce se poate forma cu toate cifrele lui n (fiecare dintre cele 10 cifre se va folosi cel mult o dată). De exemplu, pentru 50336 se va afişa 3056.

11. Se dă o matrice de numere întregi. Să se afişeze elementele comune:
a) tuturor liniilor matricei; b) tuturor coloanelor matricei.

12. Se dă un text cu cel mult 255 de caractere. Să se afişeze caracterele diferite din text de pe poziţiile pare. De exemplu, pentru textul „`Abracadabra`" se va afişa „`abr`".

13. Să se afişeze la ecran toate submulţimile mulţimii:
a) {a, b, c, d, e}; b) {1, 2, 3, 4, 5, 6, 7}; c) {1, 2, 3, *A*, *B*, *C*}.

14. Se dă un text. Să se afişeze:
a) vocalele care se repetă în text;
b) consoanele care se repetă în text;
c) cifrele care se repetă exact de 3 ori în text.

15. Să se formuleze problema care se rezolvă cu ajutorul algoritmului:

```
program App_set;
uses Crt;
var c: set of 0..9;
    i:integer;
    n,r:longint;
BEGIN
  ClrScr;
  write('Introdu n: ');
  readln(n);
  readkey;
  c:=[];r:=0;
  while n>0 do begin
     c:=c+[n mod 10];
     n:=n div 10;
  end;
  for i:=9 downto 0 do
     if i in c then r:=10*r+i;
  write('Raspuns: ',r);
  readkey;
END.
```

C

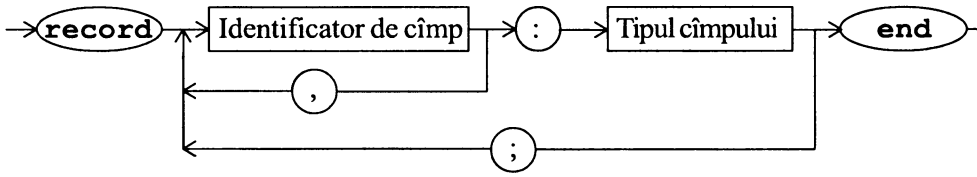
16. Se dă un text format din diferite litere şi din caracterele +, –, *, /. Să se verifice dacă textul dat reprezintă o expresie matematică. De exemplu, pentru `'a*b+c-x'` se va afişa `'Corect'`, iar pentru `'a*b+c--+x'` se va afişa `'Incorect'`.

17. Să se afişeze toate submulţimile mulţimii {1, 2, 3, ..., 50}, care au suma elementelor numărul natural dat n, $n < 1000$.

9 Tipul înregistrare (Record)

Sugestii teoretice

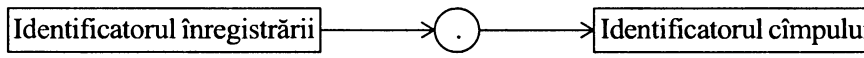
Declararea tipului record



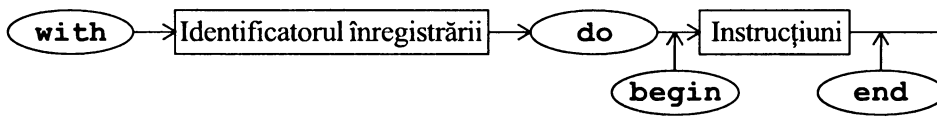
Exemple:

```
var student: record
               nume,prenume:string [20];
               anul:1..5;
               virsta:byte;
               sex:char; {m-masculin,f-feminin}
               n1,n2,n3:0..10; {notele la sesiune}
               bursa:real;
             end;
    nr_comp: record
               Re,Im:real;
             end;
```

Apelarea specificatorului de cîmp



Instrucțiunea With



În cadrul instrucțiunii **With** (și numai aici) identificatorii cîmpurilor înregistrării se vor utiliza fără a fi calificați, adică apelarea lor nu va mai include și identificatorul înregistrării.

Cuvintele-cheie **begin** și **end** se vor utiliza în cazul instrucțiunii compuse (adică în cazul în care vor fi cel puțin 2 instrucțiuni).

Probleme rezolvate

❶ Se dau două date calendaristice, reprezentînd zilele de naştere a doi prieteni, Radu şi Mihai. Să se afişeze numele celui mai mare.

Rezolvare:

Vom citi datele de naştere în două variabile de tip record.

```
program Rec1;
uses Crt;
var R,M: record
           zi:1..31;
           luna:1..12;
           an:Word;
         end;
BEGIN
   ClrScr;
   {Citim data nasterii lui Radu}
   repeat
      write('Introdu ziua, luna, anul nasterii lui Radu: ');
      readln(r.zi,r.luna,r.an);
   until (r.zi in [1..31]) and (r.luna in [1..12]) and (r.an<2000);
   {Citim data nasterii lui Mihai folosind instructiunea With}
   with m do
      repeat
         write('Introdu ziua, luna, anul nasterii lui Mihai: ');
         readln(zi,luna,an);
      until (zi in [1..31]) and (luna in [1..12]) and (an<2000);
   {Determinam cine e mai mare}
   if (r.an>m.an) or ((r.an=m.an) and (r.luna>m.luna)) or
      ((r.an=m.an) and (r.luna=m.luna) and (r.zi>m.zi) then
      write('Mihai');
   if (m.an>r.an) or ((m.an=r.an) and (m.luna>r.luna)) or
      ((m.an=r.an) and (m.luna=r.luna) and (m.zi>m.zi)) then
      write('Radu');
   if (m.an=r.an) and (m.luna=r.luna) and (m.zi=r.zi) then
      write('Sint nascuti in aceeasi zi');
   readkey;
END.
```

❷ Să se elaboreze un algoritm care va efectua adunarea a două fracţii.

Rezolvare:

Fiecare fracţie se va citi ca un string, în care numitorul se va delimita de numărător prin simbolul „/”.

```
program Rec2;
uses Crt;
type frac=record
     numar:integer;
     numit:word;
end;
var f1,f2,f3:frac; {f3 este suma lui f1 si f2}
    s:string;
    p,cod,d:integer;
```

```
BEGIN
   ClrScr;
   writeln('Introdu prima fractie, delimitind numitorul de numarator
      prin / ');
   readln(s);
   p:=pos('/',s);
   val(copy(s,1,p-1), f1.numar, cod);{citeste numaratorul primei fractii}
   val(copy(s,p+1,length(s)-p),f1.numit,cod);
   writeln('Introdu fractia a II-a');
   readln(s);
   p:=pos('/',s);
   val(copy(s,1,p-1),f2.numar,cod);
   val(copy(s,p+1,length(s)-p),f2.numit,cod);
   with f3 do begin
      numit:=f1.numit*f2.numit;
      numar:=f1.numar*f2.numit+f2.numar*f1.numit;
      {Simplificarea fractiei}
      p:=2;d:=1;{d va fi c.m.m.d.c. al numaratorului si numitorului}
      if numar>numit then cod:=numar else cod:=numit;
      repeat
         if (numar mod p =0) and (numit mod p=0) then d:=p;
         inc(p);
      until (p>cod div 2);
      numar:=numar div d;
      numit:=numit div d;
      writeln('Suma: ',numar,'/',numit);
   end;
   readkey;
END.
```

❸ Pentru o grupă de studenţi se cunoaşte:

- numărul total de studenţi;
- numele, prenumele fiecărui student;
- 3 note la o sesiune ale fiecărui student.

a) Să se calculeze bursa (b) fiecărui student conform formulei:

$$b = \begin{cases} 0, \text{ dacă } n_m < 6 \\ 100 \text{ lei, dacă } 6 \le n_m < 7 \\ 20 * n_m, \text{ dacă } n_m \ge 7, \end{cases}$$

unde n_m este nota medie a studentului.

b) Să se afişeze lista studenţilor restanţieri (care au cel puţin una din note mai mică decît 5).

Rezolvare:

Vom păstra lista studenţilor într-un vector ale cărui componente sînt înregistrări.

```
program Rec2;
uses Crt;
type st=record
          nume,prenume:string[20];
          n1,n2,n3:0..10;
          nm,b:real; {nm - nota medie, b - bursa}
       end;
```

```
var lista:array[1..30] of st;
    i,n:byte; {n - numarul de studenti}
BEGIN
   ClrScr;
   write('Numarul de studenti: ');
   readln(n);
   for i:=1 to n do begin
      writeln('Studentul ',i);
      with lista[i] do begin
         write('Nume: '); readln(nume);
         write('Prenume: '); readln(prenume);
         write('Notele: '); readln(n1,n2,n3);
         nm:=(n1+n2+n3)/3;
         if (nm>=6) and (nm<7) then b:=100;
         if nm>=7 then b:=nm*20;
      end;
   end;
   writeln('Restantieri: ');
   for i:=1 to n do
      with lista[i] do
         if (n1<5) or (n2<5) or (n3<5) then writeln(Nume,' ',Prenume);
   readkey;
END.
```

❹ Se citesc datele despre candidaţii la admiterea într-o instituţie de invăţământ. Pentru fiecare candidat se cunoaşte:
- numele şi prenumele;
- data naşterii (ziua, luna, anul);
- sexul;
- studii (medii sau liceu).

Pentru fiecare absolvent de liceu se ştie nota medie din diplomă. Fiecare absolvent al şcolii medii susţine 2 examene (la română şi la profil), după care i se calculează nota medie. Să se afişeze lista candidaţilor în ordinea descreşterii notei medii.

Rezolvare:

```
program Rec4;{inregistrari cu variante}
uses Crt;
type TStudii=(bac,medii);
     data=record
             zi:1..31;
             luna:1..12;
             an:word;
          end;
     TCand=record {inregistrare ierarhizata si cu variante}
              nume,pren:string[20];
              dn:data;
              sex:char;
              nm:real;
              case studii:Tstudii of
                 medii:(rom,profil:1..10);
                 bac:();
           end;
```

```
var a:array[1..50] of Tabit;{vector cu inregistrari}
    t:TCand;
    n,i,j:integer;
    s:char;
BEGIN
   ClrScr;
   write('Introdu numarul de candidati: ');
   readln(n);
   for i:=1 to n do
      with a[i] do begin
         writeln(' ------ ',i,' ------');
         write('Nume: '); readln(nume);
         write('Prenume: '); readln(pren);
         write('Data, luna, anul nasterii: ');
         readln(dn.zi,dn.luna,dn.an);
         write('Sexul(m/f): '); readln(sex);
         write('Bacalaureat (b) sau scoala medie (m): ');
         readln(s);
         if s='m' then begin
            studii:=medii;
            write('Romana: ');
            readln(rom);
            write('Profil: ');
            readln(profil);
            nm:=(rom+profil)/2;
         end
         else begin
            studii:=bac;
            write('Nota medie: ');
            readln(nm);
         end;
         writeln;
      end;
   {Ordonarea listei}
   for j:=1 to n do
      for i:=1 to n-1 do
         if a[i].nm<a[i+1].nm then begin
            t:=a[i];
            a[i]:=a[i+1];
            a[i+1]:=t;
         end;
   {Afisarea listei}
   for i:=1 to n do
      with a[i] do begin
         write(nume:15,' ',pren:15,' ',sex:3,' ',nm:6:2);
         if studii=medii then
            write(' {Rom: ',rom:2,' Profil: ',profil:2,'}');
         writeln;
      end;
   readkey;
END.
```

Observaţie

Tipul `TCand` este înregistrare cu variante. El conţine cîmpul `Studii` (de tip enumerare), numit **cîmp selector**. Numărul şi tipurile cîmpurilor specificate după cîmpul selector depind de valoarea curentă a acestuia. Astfel, în cazul valorii `bac` înregistrarea nu va mai avea nici un cîmp, iar în cazul valorii `medii` vor urma cîmpurile `rom` şi `profil`. Partea cu variante se scrie întotdeaună ultima. Cîmpul selector poate fi de orice tip ordinal.

Probleme propuse

A

1. Utilizînd tipul de date record, să se realizeze un algoritm pentru efectuarea operaţiilor aritmetice asupra a două numere complexe.

2. Utilizînd tipul de date record, să se realizeze un algoritm pentru efectuarea operaţiilor aritmetice (adunarea, scăderea, înmulţirea, împărţirea) asupra a două fracţii.

3. Se dau 3 date calendaristice, reprezentînd zilele de naştere a trei prieteni: Sergiu, Ion şi Andrei. Să se afişeze numele celui mai în vîrstă.

4. Utilizînd tipul de date record, să se realizeze un algoritm pentru calcularea vîrstei în ani a unei persoane, fiind date:
   - data naşterii (ziua, luna, anul);
   - data curentă (ziua, luna, anul).

5. Se introduce de la tastatură:
   - data calendaristică curentă (ziua, luna);
   - denumirea unei zile a săptămînii.

   Să se calculeze cîte zile cu această denumire au fost de la începutul anului curent pîna în ziua curentă.

6. Se introduc de la tastatură două date calendaristice (data, luna, anul). Să se calculeze diferenţa de zile dintre aceste date.

7. Se introduc de la tastatură:
   - două date calendaristice (ziua, luna);
   - denumirea unei zile a săptămînii.

   Să se calculeze cîte zile cu această denumire au fost de la o dată pînă la alta.

8. Se introduce de la tastatură:
   - o dată calendaristică (ziua, luna, anul);
   - un număr natural *n*.

a) Să se afişeze data calendaristică care va fi peste n zile.
b) Să se afişeze data calendaristică care a fost cu n zile în urmă.

9. Se dau trei date calendaristice, reprezentînd zilele de naştere a trei prieteni: Sergiu, Ion şi Andrei. Să se afişeze numele celui mijlociu.

10. Pentru o grupă de studenţi se cunoaşte:
- numărul de studenţi;
- numărul de examene la sesiune;
- numele şi prenumele fiecărui student;
- sexul fiecărui student;
- data naşterii fiecărui student;
- notele la o sesiune a fiecărui student.

a) Să se afişeze lista restanţierilor.
b) Să se afişeze lista studenţilor în ordinea alfabetică a numelui (ordine lexicografică).
c) Să se calculeze bursa fiecărui student după formula:
- 0 lei, dacă nota medie (n_m) este mai mică decît 7;
- 100 lei, dacă $7 \le n_m < 8{,}5$;
- $20 \cdot n_m$ lei, dacă $n_m \ge 8{,}5$.

d) Să se afişeze lista fetelor care au 20 de ani împliniţi.
e) Să se afişeze numele şi prenumele studentului (eventual studenţilor) cu cea mai mare notă medie.
f) Să se afişeze prenumele care se repetă în grupă.
g) Să se calculeze procentul băieţilor.
h) Să se afişeze prenumele cel mai des întîlnit în grupă.

11. Pentru un eşantion social de persoane se cunoaşte:
- numărul pesoanelor;
- vîrsta fiecărei persoane;
- înălţimea fiecărei persoane;
- masa (greutatea) fiecărei persoane;
- sexul fiecărei persoane;
- starea civilă (căsătorită sau nu).

a) Să se determine procentul persoanelor sub 20 de ani.
b) Să se determine procentul persoanelor cu înălţimea mai mare de 170 cm.
c) Să se determine masa medie a unei persoane de peste 18 ani.
d) Să se determine ce procent din numărul persoanelor de sex feminin au peste 20 de ani şi nu sînt căsătorite.
e) Să se determine ce procent din numărul persoanelor între 20 şi 50 de ani au greutatea mai mare decît greutatea medie.

12. Pentru o listă de paralelograme se cunoaşte:
- denumirea fiecărui paralelogram (de exemplu, *ABCD*, *MNKP*);

- dimensiunile fiecărui paralelogram;
- măsura unui unghi al fiecărui paralelogram.

a) Să se determine tipul fiecărui paralelogram (arbitrar, dreptunghi, romb, pătrat).
b) Să se determine perimetrul şi aria fiecărui paralelogram.
c) Să se afişeze denumirea şi diagonalele fiecărui paralelogram.

13. Pentru o listă de triunghiuri se cunoaşte:
- denumirea fiecărui triunghi (de exemplu, *ABC*, *MNK*);
- lungimile laturilor fiecărui triunghi;
- măsura a două unghiuri ale fiecărui triunghi.

a) Să se determine tipul fiecărui triunghi (scalen, dreptunghic, ascuţitunghic, obtuzunghic, echilateral, isoscel).
b) Să se determine perimetrul şi aria fiecărui triunghi.

14. Utilizînd tipul de date record, să se descrie o agendă de telefoane pentru care se cunoaşte:
- numărul de abonaţi;
- numele, prenumele fiecărui abonat;
- numărul de telefon al fiecărui abonat;
- adresa (strada, numărul casei) fiecărui abonat.

a) Să se afişeze lista abonaţilor, al căror număr de telefon începe cu 47.
b) Să se afişeze numele, prenumele şi numărul de telefon pentru fiecare abonat de pe strada dată.
c) Să se determine prenumele şi numărul de telefon pentru abonaţii cu numele dat.

15. Să se descrie, utilizînd tipul de date record, o bază de date despre o bibliotecă. Se cunoaşte:
- numărul total de cărţi;
- denumirea fiecărei cărţi;
- numele, prenumele primului autor al fiecărei cărţi;
- numărul de pagini ale fiecărei cărţi;
- tematica fiecărei cărţi (roman, manual, poezii etc.);
- limba în care este scrisă cartea;
- editura la care a apărut cartea;
- ţara în care s-a editat cartea;
- anul ediţiei fiecărei cărţi.

a) Să se afişeze lista cărţilor autorului dat.
b) Să se afişeze lista cărţilor apărute la editura dată.
c) Să se afişeze lista cărţilor editate în limba română peste hotarele ţării.
d) Să se afişeze lista cărţilor editate la tema dată după anul 2000.
e) Să se afişeze tema la care sînt editate cele mai multe cărţi.

16. Pentru o listă de produse alimentare dintr-un magazin se cunoaşte:
- numărul total de produse;
- denumirea fiecărui produs;

- data fabricării fiecărui produs;
- data expirării valabilității fiecărui produs;
- prețul inițial al fiecărui produs;
- prețul actual al fiecărui produs.

Prețul actual depinde de data curentă:
- dacă ea a depășit data expirării termenului de valabilitate a produsului, atunci prețul actual este 0;
- dacă s-a ajuns la mijlocul termenului de valabilitate, atunci prețul scade cu 20% față cel inițial;
- dacă pînă la expirarea termenului de valabilitate a mai rămas cel mult 0,25 din acest termen, atunci prețul actual este cu 50% mai mic decît cel inițial.

Știind data curentă, să se afișeze:
a) lista produselor cu termenul de valabilitate expirat;
b) lista produselor cu o reducere la preț de 50%;
c) lista produselor cu o reducere la preț de 20%;
d) lista produselor cu termenul de valabilitate de cel puțin 1 an;
e) lista produselor cu termenul da valabilitate de cel mult o lună.

17. Pentru o listă de programe TV se știe:
- numărul total de emisiuni;
- denumirea fiecărei emisiuni;
- canalul pe care va rula emisiunea;
- tipul fiecărei emisiuni (film artistic, divertisment, știri, desene animate);
- începutul fiecărei emisiuni;
- sfîrșitul fiecărei emisiuni.

Să se afișeze:
a) lista emisiunilor unui canal dat;
b) lista filmelor artistice;
c) lista emisiunilor de divertisment ale unui canal dat;
d) lista desenelor animate difuzate între orele 15:00 și 19:00;
e) lista filmelor artistice cu durata mai mare de 1 oră și 45 de minute;
f) numărul de știri pentru fiecare canal;
g) lista tuturor emisiunilor grupate pe canale.

B

18. Utilizînd tipul de date record, să se scrie un algoritm pentru adunarea a două polinoame de o singură nedeterminată.

19. Utilizînd tipul de date record, să se scrie un algoritm pentru înmulțirea a două polinoame de o singură nedeterminată.

20. Se realizează un concurs-sondaj în vederea stabilirii celei mai populare melodii din țară. Se cunoaște:
- numărul total de intervievați;

- numele fiecărui intervievat;
- sexul fiecărui intervievat;
- vîrsta fiecărui intervievat;
- 3 melodii dintr-o listă dată de şlagăre în ordinea preferinţelor pentru fiecare intervievat.

a) Să se afişeze lista primelor 3 melodii în ordinea popularităţii.

b) Fiecare persoană intervievată va fi punctată cu:
- 10 puncte pentru fiecare melodie pentru care a ghicit poziţia în topul stabilit în a);
- 5 puncte pentru fiecare melodie dacă a greşit cu o poziţie;
- 3 puncte pentru fiecare melodie dacă a greşit cu două poziţii.

Să se afişeze datele despre persoanele care au ocupat primele 5 locuri.

21. O casă de schimb valutar a stabilit următoarele cursuri:
- 1$ = 12 lei;
- 1 euro = 16 lei;
- 1 rublă = 0,5 lei.

În urma fiecărei tranzacţii ($x \longleftrightarrow$ lei sau lei $\longleftrightarrow x$) se înregistrează următoarele informaţii:
- data, luna, anul tranzacţiei;
- denumirea valutei încasate;
- suma încasată.

Să se efectueze următoarele bilanţuri:

a) Suma comisionului total (în lei), dacă la fiecare tranzacţie se percep 2% din sumă.

b) Valuta străină cea mai solicitată (pentru care au fost încheiate cele mai multe tranzacţii).

c) Data, luna, anul realizării celei mai avantajoase tranzacţii (cea mai mare sumă încasată).

22. Timpul (condiţiile meteorologice) pentru o zi poate fi considerat o dată de tipul

```
type Timp=record
            temperatura:integer;
            umiditatea:integer;
            case vint:boolean of
               true:(directie: (nord, sud, est, vest, nordvest,
                               sudest, nordvest, sudvest)
                     viteza:integer);
               false:();
         end;
```

Se citeşte de la tastatură timpul pentru 10 zile.

Să se afişeze la ecran:

a) temperatura medie;

b) numărul zilelor fără vînt;

c) ziua cînd temperatura a fost cea mai mică;

d) ziua cînd a fost vînt cu viteză maximă;

e) direcţia din care cel mai des a bătut vîntul;

f) valoarea umidităţii în zilele cînd vîntul a bătut din est.

23. La o fabrică de băuturi alcoolice se cunoaşte:
- numărul total de produse;
- tipul fiecărui produs (vin, coniac, şampanie);
- denumirea fiecărui produs;
- vîrsta fiecărui produs;
- culoarea fiecărui produs;
- procentul de alcool al fiecărui produs;
- procentul de zahăr al fiecărui produs;
- soiurile de struguri folosite la fabricarea fiecărui produs (dintr-o listă dată: de exemplu, Sauvignon, Cabernet, Izabela, Pinot, Traminer, Chardonney, Moldova, Merlot etc.);
- preţul unui litru de produs.

Să se afişeze:
a) lista vinurilor seci (procentul de alcool mai mic de 15%);
b) lista produselor tari (peste 30% de alcool) cu vîrsta mai mare de 5 ani;
c) lista vinurilor, la producerea cărora nu au fost folosite soiurile Izabela şi Moldova;
d) lista produselor în ordinea descreşterii preţurilor;
e) cel mai scump produs-şampanie;
f) cel mai ieftin produs-coniac;
g) soiul de poamă cel mai des folosit la fabricarea vinurilor;
h) lista vinurilor mai scumpe decît cel puţin un produs-coniac.

24. Se dă una din următoarele figuri geometrice: cerc, triunghi, dreptunghi. Fiecare figură se dă diferit:
- în cazul cercului se dă raza lui;
- în cazul triunghiului – laturile lui;
- în cazul dreptunghiului – dimensiunile lui.

În funcţie de figura dată să se afişeze o informaţie anumită despre ea:
- în cazul cercului – lungimea lui;
- în cazul triunghiului – aria lui;
- în cazul dreptunghiului – lungimea diagonalei lui.

Indicaţie. Se va declara tipul înregistrare cu variante *Figura* care va conţine un cîmp-selector *TipFigura*.

C

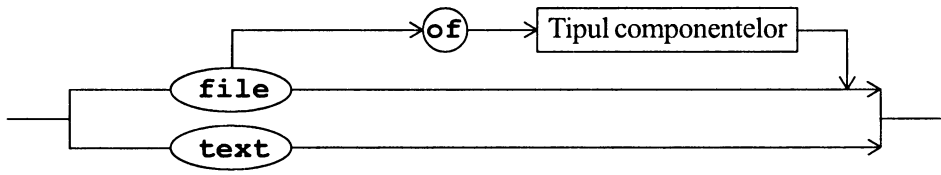
25. Utilizînd tipul de date înregistare, să se scrie un algoritm pentru înmulţirea a două polinoame de mai multe nedeterminate.

26*. Utilizînd tipul de date înregistare, să se scrie un algoritm pentru împărţirea a două polinoame:
a) de o singură nedeterminată;
b) de mai multe nedeterminate.

10 Tipul fişier

Sugestii teoretice

Declararea tipului fişier



Exemple:

```
type  student=record
         nume,prenume:string [20];
         an,nota1,nota2:byte;
      end;
var f:file of integer; {fisier ale carui componente sint numere intregi}
    g:file of student; {fisier ale carui componente sint inregistrari de
      tip student}
    t:text; {fisier text ale carui componente sint caractere si marcatori
      de sfirsit de linie}
    h:file; {fisier (netipizat) pentru care nu se declară din timp tipul
      componentelor}
```

Observaţii

1. Tipul componentelor poate fi oricare tip admis de Turbo Pascal, cu excepţia tipului fişier sau a unui tip ce conţine tipul fişier.

2. În Turbo Pascal fişierele *text* sînt fişiere secvenţiale (componentele lor pot fi accesate numai în ordinea în care ele apar în cadrul fişierului).

3. Fişierele tipizate (declarate cu **`file of`**) sînt fişiere cu acces direct, în sensul că Turbo Pascal conţine funcţii şi proceduri pentru accesarea directă a oricărei componente *i* din fişier. Fără utilizarea acestor funcţii, accesul la componente se realizează secvenţial, adică după citirea/scrierea unei componente, poziţia curentă de fişier se deplasează la următoarea componentă.

Prima componentă din fişier are poziţia 0, cea de-a doua – poziţia 1 ş.a.m.d.

4. Fişierele *text* se reprezintă pe suporturi, sub formă externă (prin secvenţe de caractere), celelalte fişiere – sub formă internă (prin secvenţe de cifre binare).

Funcții și proceduri standard pentru lucrul cu fișierele

Procedura **Assign**(*f*, *nume*) asociază fișierul extern *nume* variabilei de tip fișier *f*.

Procedura **Rewrite**(*f*) pregătește pentru (re)scriere fișierul *f*, înlocuindu-l cu un fișier vid (care nu conține nici o componentă).

Procedura **Write**(*f*, *x*) adaugă în fișierul *f* o componentă, și anume, valoarea returnată de expresia *x*. Evident, *x* trebuie să aibă tipul compatibil cu tipul componentelor lui *f*.

Procedura **Reset**(*f*) pregătește fișierul *f* pentru citire, mutînd poziția de citire la începutul fișierului.

Procedura **Read**(*f*, *v*) atribuie următoarea componentă a fișierului *f* variabilei *v*, după care avansează poziția de citire dincolo de această componentă.

Observație

Secvența de instrucțiuni `write(`f, x_1`)`, `write(`f, x_2`)`, ..., `write(`f, x_n`)` (respectiv `read(`f, v_1`)`, `read(`f, v_2`)`, ..., `read(`f, v_n`)`) este echivalentă cu instrucțiunea `write(`$f, x_1, x_2, ..., x_n$`)` (respectiv cu `read(`$f, v_1, v_2, ..., v_n$`)`).

Funcția **Eof**(*f*) returnează valoarea *true*, dacă poziția de citire în *f* este sfîrșit de fișier, în caz contrar – valoarea *false*.

Procedura **Close**(*f*) închide fișierul *f*.

Procedura **Rename**(*f*, *nume_nou*) redenumește numele fișierului asociat variabilei *f* prin *nume_nou*.

Procedura **Erase**(*f*) șterge fișierul *f*. Fișierul trebuie să fie închis.

Procedura **ChDir**(*nume_catalog*). În urma realizării procedurii directoriul *nume_catalog* devine curent. String-ul *nume_catalog* conține și calea pînă la directoriu.

Procedura **GetDir**(*d*, *s*), unde *d* este de tip `word`, iar *s* de tip `string`, atribuie variabilei *s* numele directoriului curent al discului *d* (1 – discul A, 2 – discul B etc.)

Procedura **MkDir**(*nume_catalog*) creează un directoriu cu numele indicat în *nume_catalog* (acesta conține și calea).

Procedura **RmDir**(*nume_catalog*) șterge directoriul vid *nume_catalog*.

Funcția **FilePos**(*f*) returnează o valoare de tip `longint`, care reprezintă numărul de ordine al componentei curente din fișierul cu acces direct *f*.

Funcția **FileSize**(*f*) returnează o valoare de tip `longint`, care reprezintă numărul total de componente ale fișierului cu acces direct *f*.

Procedura **Seek**(*f*, *i*) mută poziția de citire/scriere pe componenta cu numărul de ordine *i* din fișierul cu acces direct *f*.

Procedura **BlockRead**(*f*, *v*, *count*[, *r*]) citește *count* componente din fișierul netipizat *f* și le atribuie variabilei *v*. Parametrul neobligatoriu *r* conține numărul efectiv de componente efectiv citite. Dacă procedura a fost executată corect, atunci *count* = *r*.

Procedura **BlockWrite**(*f*, *v*, *count*[, *r*]) transmite *count* componente din variabila *v* în fișierul *f*. Parametrul neobligatoriu *r* conține numărul componentelor efectiv transmise. Dacă procedura a fost executată corect, atunci *count* = *r*.

Funcția **Eoln**(*f*) returnează valoarea *true*, dacă s-a ajuns la sfîrșit de linie într-un fișier *text*.

Observații

1. Funcțiile `FileSize`, `FilePos` și procedura `Seek` se utilizează doar în fișiere cu acces direct (tipizate).

2. Procedurile `BlockRead` și `BlocWrite` se utilizează doar în fișiere netipizate (fără tip).

Probleme rezolvate

❶ a) Să se scrie un algoritm care citește de la tastatură datele despre o grupă de studenți. Pentru fiecare student se cunoaște numele, prenumele, 3 note la ultima sesiune. Datele citite se stochează într-un fișier.

b) Să se scrie un algoritm care va citi datele din fișierul creat în a) și va afișa la ecran numele și prenumele studenților care au cel puțin o restanță.

Rezolvare:

a) Vom păstra datele într-un fișier, ale cărui componente vor fi de tipul înregistrare.

```
program File1a;
uses Crt;
type  Student=record
         nume,prenume:string[20];
         n1,n2,n3:0..10;
      end;
var f:file of student;
    v:student;
    i,n:byte; {n - numarul de studenti}
BEGIN
   ClrScr;
   write('Numarul de studenti: ');
   readln(n);
   assign(f,'c:\grupa'); {asociem fisierul 'grupa' de pe discul c:
      variabilei f}
   rewrite(f);
   for i:= 1 to n do begin
      writeln('Studentul ',i);
      with v do begin
         write('Nume: '); readln(nume);
         write('Prenume: '); readln(prenume);
         write('Notele: '); readln(n1,n2,n3);
      end;
      write(f,v);
   end;
   close(f);
   readkey;
END.
```

b)
```
program File1b;
uses Crt;
type  student=record
         nume,prenume:string[20];
         n1,n2,n3:0..10;
      end;
var f:file of student;
    v:student;
    i:byte;
BEGIN
   ClrScr;
   assign(f,'c:\grupa');
   reset(f);
   writeln('Restantieri:');
   while not eof(f) do begin
      read(f,v);
      with v do
         if (n1 in [0..4]) or (n2 in [0..4]) or
            (n3 in [0..4]) then writeln(Nume,' ',prenume);
   end;
   close(f);
   readkey;
END.
```

❷ Să se scrie un algoritm care citeşte de la tastatură *n* rînduri de caractere şi le scrie într-un fişier *text*. Numărul natural *n* este dat.

Rezolvare:

```
program File2;
uses Crt;
var f:text;
    s:string;
BEGIN
   ClrScr;
   write('Numarul de rinduri: ');
   readln(n);
   assign(f, 'c:\proba.txt');
   rewrite(f);
   for i:=1 to n do begin
      readln(s);
      writeln(f,s);
   end;
   close(f);
   readkey;
END.
```

❸ Să se scrie un algoritm care va efectua adunarea a două polinoame de mai multe nedeterminate.

Polinoamele pot fi oricît de mari.

Rezolvare:

Putem organiza polinoamele (acestea fiind alcătuite din monoame) ca vectori, ale căror componente vor fi înregistrări (primul cîmp – coeficientul monomului, al doilea – partea lui

laterală), însă în Turbo Pascal vectorii nu pot fi prea mari (cel mult circa 200 de componente). De aceea vom păstra fiecare polinom de intrare şi polinomul rezultat într-un fişier *text* cu următoarea structură a conţinutului:

- fiecare monom va fi scris din rînd nou;
- coeficientul se va scrie din poziţia 1, iar partea literală – începînd cu poziţia 10,
- fiecare nedeterminată va fi scrisă de un număr de ori, egal cu exponentul puterii acestei nedeterminate.

De exemplu, pentru polinomul $12ax^3y + 214axy^4 - 9x^3y + 25ab - 8$ vom avea următorul conţinut de fişier:

12	*axxxy*
214	*axyyyy*
–9	*xxxy*
25	*ab*
–8	

Programul va consta din două etape:

La etapa I fiecare monom din primul polinom se va aduna cu monoamele asemenea din polinomul al doilea. Rezultatul se va scrie în fişierul rezultat.

La etapa a II-a monoamele din polinomul al doilea, care nu au fost adunate cu monoamele primului polinom, vor fi ataşate la rezultatul obţinut la prima etapă.

Exemplu

$$\begin{array}{l} 3ax^2 - 2xy + 5ay + 3 \\ + \quad 2ax^2 - 3x^2y + 6ay + 2xy \\ \hline 5ax^2 + 11ay + 3 - 3x^2y \end{array}$$

polinomul 1 (p1.txt)

3	*axx*
–2	*xy*
5	*ay*
3	

polinom 2 (p2.txt)

2	*axx*
–3	*xxy*
6	*ay*
2	*xy*

polinomul rezultat (p3.txt)

După I etapă		După etapa a II-a	
5	*axx*	5	*axx*
11	*ay*	11	*ay*
3		3	
		–3	*xxy*

```
program File3;
uses Crt;
var f1,f2,f3:text;
    s1,s2,c1,c2,text1,text2:string;
    i1,i2,j,code1,code2,coef1,coef2:integer;
    f:boolean;
BEGIN
  ClrScr;
  assign(f1,'c:\p1.txt');
  assign(f2,'c:\p2.txt');
  assign(f3,'c:\p3.txt');
```

```
reset(f1);
rewrite(f3);
while not eof(f1) do begin
   readln(f1,s1); {citeste un monom din primul polinom}
   while s1[length(s1)]=' ' do s1:=copy(s1,1,length(s1)-1);
   {sterge spatiile de la sfirsit}
   i1:=1;
   c1:='';
   repeat
      c1:=c1+s1[i1]; {c1 - coeficientul primului polinom}
      inc(i1);
   until s1[i1]=' ';
   text1:=copy(s1,10,length(s1)-9);
   val(c1,coef1,code1);
   reset(f2);
   text2:='';
   while (not eof(f2)) and (text1<>text2) do begin
      readln(f2,s2); {citeste un monom din polinomul 2}
      while s2[length(s2)]=' ' do s1:=copy(s2,1,length(s2)-1);
      {sterge spatiile de la sfirsit}
      i2:=1;
      c2:='';
      repeat
         c2:=c2+s2[i2];
         inc(i2);
      until s2[i2]=' ';
      text2:=copy(s2,10,length(s2)-9);
      val(c2,coef2,code2);
      if text2=text1 then begin
         coef1:=coef1+coef2;
         str(coef1,c1);
         s1:=c1;
         for j:=1 to 9-length(s1) do s1:=s1+' ';
         s1:=s1+text2;
      end;
   end;
   if s1[1]<>'0' then writeln(f3,s1);
   close(f2);
end;
close(f1);
{================= Etapa 2 =================}
reset(f2);
while not eof(f2) do begin
   readln(f2,s2);
   while s2[length(s1)]=' ' do s2:=copy(s2,1,length(s2)-1);
   {sterge spatiile de la sfirsit}
   i2:=1;
   c2:='';
   repeat
      c2:=c2+s2[i2];
      inc(i2);
   until s2[i2]=' ';
   text2:=copy(s2,10,length(s2)-9);
```

```
      f:=false;
      text1:='';
      reset(f1);
      while (not eof(f1)) and (not f) do begin
         readln(f1,s1);
         while s1[length(s1)]=' ' do s1:=copy(s1,1,length(s1)-1);
         {sterge spatiile de la sfirsit}
         i1:=1;
         c1:='';
         repeat
            c1:=c1+s1[i1];
            inc(i1);
         until s1[i1]=' ';
         text1:=copy(s1,10,length(s1)-9);
         if text2=text1 then f:=true;
      end;
      if not f then writeln(f3, s2);
      close(f1);
   end;
   close(f2);
   close(f3)
END.
```

❹ Să se scrie un algoritm care va realiza copierea conţinutului unui fişier într-un alt fişier.

Rezolvare:

```
program File4; {algoritm cu fisiere fara tip}
uses Crt;
var sursa,dest:string;
    buf:array[1..1000] of char;
    rez:integer;
    fs,fd:file;
BEGIN
   ClrScr;
   write('Scrie numele fisierului sursa: ');
   readln(sursa);
   assign(fs,sursa);
   reset(fs,1);
   write('Scrie numele fisierului destinatie: ');
   readln(dest);
   assign(fd,dest);
   rewrite(fd,1);
   BlockRead(fs,buf,sizeof(buf),rez);
   while rez>0 do begin
      BlockWrite(fd,buf,rez);
      BlockRead(fs,buf,sizeof(buf),rez);
   end;
   close(fs);
   close(fd);
   write('Ok!!!');
   readkey;
END.
```

❺ Să se afişeze la ecran denumirile fişierelor cu algoritmi Pascal (fişiere .pas). Pentru fiecare denumire utilizatorul va putea viziona conţinutul fişierului respectiv.

Rezolvare:

```
program File5;
uses Crt,Dos;
var fisier:searchrec;
    tasta:char;
procedure afisare(nume:string);{afisarea continutului fisierului la ecran}
var f:text;
    r:string;
    i:integer;
begin
   i:=1;
   assign(f,nume);
   reset(f);
   while not eof(f) do begin
      inc(i);
      if i mod 24=0 then readkey; {dupa fiecare 24 de linii se va opri}
      readln(f,r);
      writeln(r);
   end;
   close(f);
end;
BEGIN
   ClrScr;
   findfirst('*.pas',anyfile,fisier); {cauta primul .pas fisier}
   while (doserror=0) and (tasta<>#27) do begin
      ClrScr;
      writeln(fisier.name:20,' Afisam continutul? (y/n) ');
      tasta:=upcase(readkey);
      if tasta='Y' then begin
         afisare(fisier.name);
         writeln;
         writeln('Apasa orice tasta');
         readkey;
      end;
      findnext(fisier); {cauta urmatorul fisier}
   end;
END.
```

Probleme propuse

A

1. Să se creeze un fişier *text* care va conţine toate literele, mari şi mici, ale alfabetului latin, cîte două în rînd: litera mică, apoi cea mare.
2. Să se creeze un fişier *text* care va conţine toate numerele naturale de la 1 la 999 (în ordine crescătoare), cîte trei în fiecare rînd.
3. Să se creeze un fişier *text* care va conţine toate numerele întregi (în ordine crescătoare) cu modulul mai mic decît 200, cîte 4 în fiecare rînd.

4. Să se creeze un fişier *text*, în care:
   primul rînd reprezintă 10 cifre „0”,
   rîndul 2 – 10 cifre „1”,
   rîndul 3 – 10 cifre „2”,
   ...
   rîndul 10 – 10 cifre „9”.

5. Să se creeze un fişier *text* din 26 de rînduri, în care:
   primul rînd reprezintă 10 litere „A”,
   rîndul 2 – 10 litere „B”,
   rîndul 3 – 10 litere „C”,
   ...
   rîndul *n* – 10 litere, care coincid cu litera a *n-a* din alfabetul latin, unde *n* este număr natural mai mic decît 27.

6. Să se creeze un fişier *text* din 20 de rînduri, în care:
   primul rînd reprezintă 10 cifre „0”,
   rîndul 2 – 10 litere „a”,
   rîndul 3 – 10 cifre „1”,
   rîndul 4 – 10 litere „b”,
   ...
   rîndul 19 – 10 cifre „9”,
   rîndul 20 – 10 litere „j”.

7. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se afişeze conţinutul lui la ecran.

8. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se afişeze la ecran rîndurile care conţin nu mai mult de 30 de caractere (inclusiv spaţiile).

9. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier, substituind fiecare literă „e” prin litera „i”.

10. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se calculeze cîte rînduri are fişierul.

11. Să se creeze un fişier *text* în care se va scrie tot ce va culege utilizatorul de la tastatură. Executarea programului (respectiv înscrierea în fişier) se va sfîrşi atunci cînd se va culege din rînd nou textul „STOP”.

12. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier, inserînd cîte 2 spaţii la începutul fiecărui rînd.

13. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier, eliminînd spaţiile de la sfîrşitul fiecărui rînd.

14. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se numere literele şi cifrele ce se conţin în fişierul dat.

15. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se afişeze la ecran rîndurile de lungime minimală.

16. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier, care va conţine rîndurile de lungime maximală ale primului fişier.

17. Se dau două fişiere *text*. Să se verifice dacă sînt identice conţinuturile acestor fişiere (simbol cu simbol). În caz negativ, să se afişeze numărul de ordine al primelor rînduri care nu coincid.

18. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier *text* în care ordinea rîndurilor este inversă (primul va deveni ultimul, iar ultimul va devenu primul; al doilea va deveni penultimul ş.a.m.d.).

19. Se dă un text şi un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se calculeze cîte rînduri din fişier conţin textul dat ca subşir.

20. Să se construiască un fişier *text* care va conţine tabla adunării de la 0 la 20. Conţinutul va arăta similar:

Tabla adunarii de la 0 la 3

+	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

21. Să se construiască un fişier *text* care va conţine tabla înmulţirii de la 0 la 20. Conţinutul va arăta similar:

Tabla inmultirii de la 0 la 3

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9

22. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se afişeze cuvîntul (eventual cuvintele) de lungime maximală.

23. a) Într-un fişier *text* să se genereze în 4 coloane 100 de numere întregi aleatoare cu modulul mai mic decît 50.

b) Să se creeze un fişier *text* care va conţine numerele din fişierul creat în a), întîi cele negative, apoi cele nenegative, toate scrise în 5 coloane.

24. Să se tabeleze într-un fişier *text* funcţia:

a) $f(x) = x^5 - e^x$ pe intervalul [0, 3] cu pasul 0,1;

b) $f(x) = e^x - \sin x$ pe intervalul [–1, 1] cu pasul 0,05.

25. Se dă un fişier *text* care conţine o listă de forma:

Denumirea mărfii *Numărul de unităţi* *Preţ-unitate*

Să se constriască alt fişier *text* care va conţine lista completă (obţinută din prima listă) de forma:

Denumirea mărfii *Numărul de unităţi* *Preţ-unitate* *Cost*

De exemplu, pentru lista

Mere	20	5
Prune	100	4
Cartofi	50	4

se va obţine următoarea listă în alt fişier:

Mere	20	5	100
Prune	100	4	400
Cartofi	50	4	200

26. Să se creeze un fişier cu tip care conţine un număr oarecare de numere întregi aleatoare de cel mult 5 cifre. Să se afişeze la ecran:

a) componenta maximală;

b) numărul de apariţii ale componentei minimale;

c) suma componentelor pozitive.

27. Să se creeze un fişier cu tip care conţine un număr oarecare de numere întregi aleatoare de cel mult 5 cifre. Să se transcrie conţinutul într-un alt fişier, în ordinea crescătoare a componentelor.

28. a) Să se creeze un fişier cu tip care va conţine informaţii (denumirea şi preţul) despre o listă de produse alimentare.

b) Să se afişeze denumirea produselor cu cel mai mic preţ.

c) Fiind dat un preţ, să se afişeze denumirile produselor cu acest preţ.

d) Să se afişeze lista produselor în ordinea lexicografică a denumirii lor.

e) Să se afişeze lista produselor în ordinea descrescătoare a preţurilor.

f) Să se păstreze lista ordonată în d) într-un alt fişier.

g) Să se insereze în fişierul creat în f) un nou produs dat respectînd ordinea.

29. a) Să se creeze un fişier cu tip care va conţine informaţii (numele, prenumele, anul naşterii, salariul, sexul) despre angajaţii unei întreprinderi.

b) Să se afişeze lista angajaţilor de sex feminin.

c) Să se calculeze salariul mediu al unui angajat.

d) Să se afişeze lista persoanelor cu salariul mai mic decît cel mediu.

e) Fiind date numele şi prenumele unui angajat, să se afişeze toată informaţia despre acest angajat.

f) Să se păstreze în alt fişier conţinutul fişierului creat în a), în ordinea lexicografică a numelui, prenumelui.

g) Să se insereze în fişierul creat în f) un nou angajat, respectînd ordinea.

30. a) Să se creeze un fişier cu tip care va conţine informaţii despre cărţi, precizîndu-se pentru fiecare dintre ele titlul, primul autor, anul apariţiei, editura şi preţul.

b) Să se afişeze lista cărţilor editate pînă în 2000.

c) Să se afişeze lista cărţilor cu preţul mai mare de 50 lei.

d) Să se scrie un algoritm care va permite să se corecteze datele componentei cu numărul de ordine dat.

e) Să se afişeze titlurile cărţilor unui autor dat.

f) Să se afişeze lista autorilor care au cărţi editate la cel puţin două edituri.

B

31. Să se formuleze problema care se rezolvă cu ajutorul algoritmului:

a)
```
program text1;
type fisier=file of char;
var  f1,f2:fisier;
     x:char;
     l,i:integer;
procedure init(var f1,f2:fisier);
begin
  assign(f1,'one.txt'); reset(f1);
  assign(f2,'one1.txt '); rewrite(f2);
end;
procedure invers(var f1,f2:fisier);
begin
  init(f1,f2);
  l:=filesize(f1);
  for i:=l-1 downto 0 do begin
     seek(f1,i);
     read(f1,x);
     write(f2,x);
  end;
  writeln;
  close(f1);
  close(f2);
end;
BEGIN
  invers(f1,f2);
END.
```

b)
```
program text2;
type fisier=file of char;
var  f1,f2,t:fisier;
     s:char;
BEGIN
  assign(f1,'fis1.txt'); reset(f1);
  assign(f2,'fis2.txt'); reset(f2);
  assign(t,'temp.txt'); rewrite(t);
```

```
  while not eof(f1) do begin
     read(f1,s);
     write(t,s);
  end;
  close(f1);close(t);
  rewrite(f1);
  while not eof(f2) do begin
     read(f2,s);
     write(f1,s);
  end;
  close(f1); close(f2);
  reset(t); rewrite(f2);
  while not eof(t) do begin
     read(t,s);
     write(f2,s);
  end;
  close(t); close(f2);
END.
```

32. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se construiască alt fişier, care nu va conţine rîndurile de lungime maximală ale primului fişier.

33. Se dă un fişier *text* care conţine o listă de persoane de forma:
Nume *Prenume*
Să se construiască alt fişier *text*, care va conţine aceeaşi listă, însă scrisă astfel:
Prenume *Nume*

34. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere. Să se afişeze cuvîntul (eventual cuvintele) care se repetă de cele mai multe ori.

35. Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere şi care reprezintă textul unui program Pascal. Să se verifice dacă urmează corect cuvintele-cheie `Repeat` şi `Until` (în sensul că în orice moment numărul de cuvinte `Until` va fi cel mult egal cu numărul de cuvinte `Repeat` şi numărul total de cuvinte `Until` va fi egal cu cel al cuvintelor `Repeat`).

36. a) Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere şi care reprezintă un şir de numere naturale separate printr-un spaţiu. Să se afişeze pentru fiecare rînd textul „DA" sau „NU", în funcţie de faptul dacă şirul este sau nu ordonat crescător.
b) Se dă un fişier *text* în care fiecare rînd conţine cel mult 255 de caractere şi care reprezintă un şir de numere naturale separate printr-un spaţiu.
Să se afişeze pentru fiecare rînd textul „DA" sau „NU", în funcţie de faptul dacă şirul este sau nu ordonat (crescător sau descrescător).

37. Se dă un fişier text în care fiecare rînd reprezintă 2 numere naturale separate printr-un spaţiu. Să se creeze alt fişier, ce inserează între cele 2 numere ale fiecărui rînd media aritmetică a acestor numere.

De exemplu, pentru conținutul primului fișier:

28 24
10 15
4 71

se va obține fișierul cu următorul conținut:

28 26 24
10 12.5 15
4 37.5 71

38. Să se creeze un fișier cu tip care conține un număr oarecare de numere întregi aleatoare de cel mult 5 cifre. Să se afișeze la ecran:
a) cea mai lungă secvență de componente impare;
b) numărul de secvențe (cel puțin două componente consecutive formează o secvență) de componente pare;
c) componenta care apare de cele mai multe ori în fișier.

39. Să se creeze un fișier cu tip care conține un număr oarecare de numere întregi.
a) Să se transcrie într-un alt fișier mai întîi toate componentele lui diferite de zero (păstrînd ordinea), apoi componentele nule.
b) Să se transcrie într-un alt fișier mai întîi toate componentele lui negative (păstrînd ordinea), apoi cele pozitive.
d) Să se transcrie într-un alt fișier mai întîi toate componentele lui pare (păstrînd ordinea), apoi cele impare.

40. Să se creeze un fișier cu tip care conține un număr oarecare de numere întregi. Să se creeze alt fișier, format din toate componentele diferite ale primului fișier.

41. Să se creeze un fișier cu tip care conține un număr oarecare de numere întregi. Să se creeze alt fișier, format din toate componentele care se repetă ale primului fișier. Componentele fișierului al doilea nu se vor repeta.

42. Să se scrie un algoritm care va realiza copierea conținutului unui fișier într-un alt fișier, ștergînd fișierul-sursă (va deplasa efectiv și va modifica numele fișierului-sursă).

43. Să se realizeze un algoritm ce va efectua unirea (concatenarea) unui număr dat de fișiere.

44. Utilizînd subprogramele uni-tului *DOS*, să se determine fișierul de lungime maximală a directoriului curent.

45. Utilizînd subprogramele unit-ului *DOS*, să se afișeze denumirile fișierelor din directoriul curent, ștergînd fișierele indicate de utilizator.

C

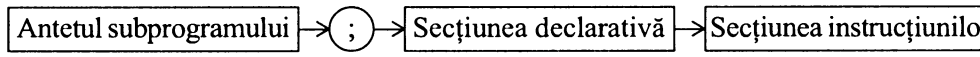
46. Să se creeze două fișiere cu tip, astfel încît fiecare să conțină un număr oarecare de numere reale ordonate crescător. Să se creeze un al treilea fișier ordonat crescător, care va conține toate numerele din primele două fișiere.

47. Să se scrie un program care va arhiva și va dezarhiva fișiere *text*. (Fișierul arhivat va ocupa mai puțină memorie decît fișierul nearhivat.)

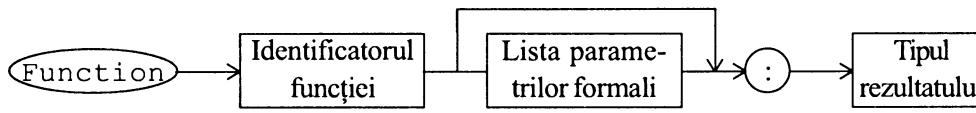
11 Subprograme

Sugestii teoretice

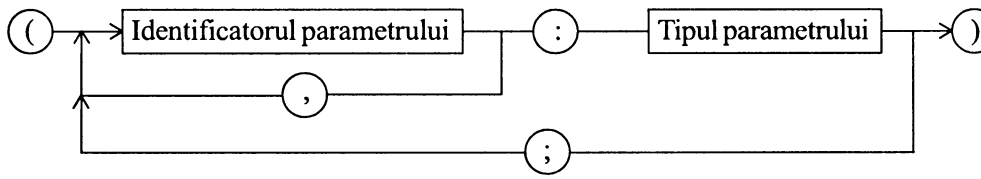
Declararea unui subprogram



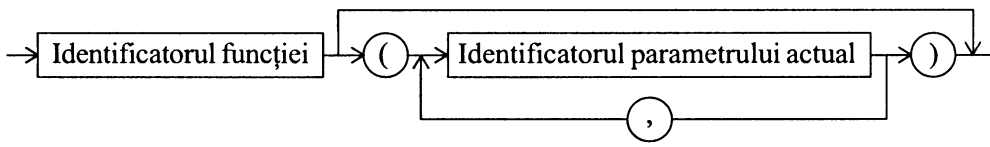
Antetul funcției



Lista parametrilor formali ai funcției



Apelul funcției



Observații

1. În calitate de parametru actual poate fi o variabilă, o expresie, un identificator de subprogram.

2. Apelarea unei funcții nu este o instrucțiune de sine stătătoare, ea trebuie inclusă ca operand în cadrul unei expresii.

3. Parametrii formali sînt disponibili numai în cadrul funcției. Numărul parametrilor actuali trebuie să fie egal cu numărul parametrilor formali din declarația funcției. Fiecare parametru actual trebuie să aibă tipul compatibil cu parametrul formal corespunzător lui.

Observații

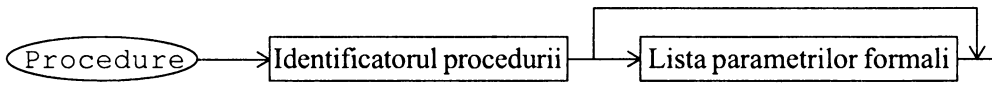
4. Rezultatul funcției este o unică valoare, deci nu poate fi structură (în afară de tipul `string`).

5. Secțiunea instrucțiunilor conține obligator ultima instrucțiune, care atribuie rezultatul numelui funcției.

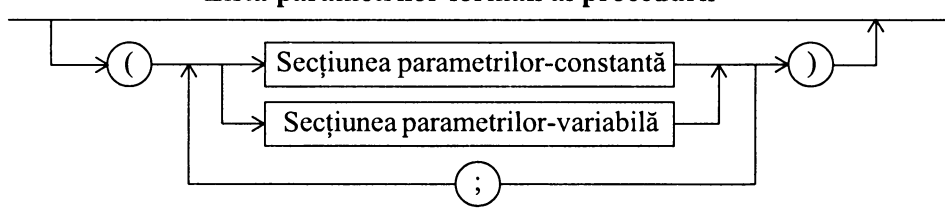
6. Subprogramele pot avea propria secțiune declarativă, unde se pot defini constante, tipuri, variabile, care pot fi utilizate doar local. De aceea, identificatorii lor se numesc **identificatori locali**.

7. Identificatorii declarați în programul principal por fi utilizați și în cadrul subprogramelor, de aceea ei se numesc **identificatori globali**.

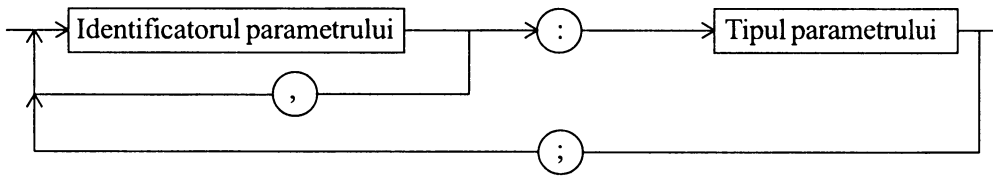
Antetul procedurii



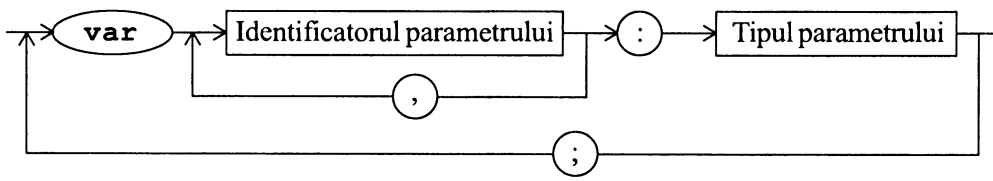
Lista parametrilor formali ai procedurii



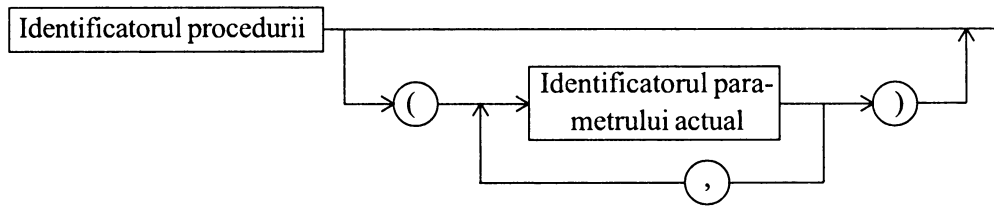
Secțiunea parametrilor-constantă



Secțiunea parametrilor-variabilă



Apelul procedurii (instrucțiunea *procedură*)



Observații

1. Procedura este un subprogram care poate returna nici una, una sau mai multe valori. Rezultatele se returnează prin intermediul parametrilor și pot fi și de tip structură.
2. Pentru proceduri datele de intrare se transmit prin **parametrii-constantă** (numiți și **parametri-valoare**) și/sau **parametrii-variabilă**. Rezultatele se returnează doar prin **parametrii-variabilă**.
3. Lista parametrilor formali poate fi vidă.
4. Parametrii formali sînt tratați ca variabile locale.

Probleme rezolvate

❶ Se dă un vector cu n $(1 \le n \le 100)$ componente întregi. Să se afișeze componentele prime.

Rezolvare:

Vom folosi procedura **Cit_tablou** pentru citirea de la tastatură a vectorului și funcția **prim** (x), care va returna o valoare de tip boolean, și anume, *true*, dacă x este prim, în caz contrar – *false*.

```
program Subpr1;
uses Crt;
type tablou=array [1..100] of integer;
var a:tablou;
    i,n:byte;
procedure cit_tablou(var t:tablou; var dim:byte);
var j:byte;
begin
   write('Dimensiunea vectorului: ');
   readln(dim);
   for j:=1 to dim do begin
      write('Componenta ',j,': ');
      readln(t[j]);
   end;
end;
function prim(x:integer):boolean;
var j:integer;
    f:boolean;
begin
   f:=true; j:=1;
```

```
    repeat
      inc(j);
      if x mod j=0 then f:=false;
    until (j>x div 2) or (not f)
    if x=1 then f:=false; {1 nu este numar prim}
    prim:=f;
end;
BEGIN
    ClrScr;
    Cit_tablou(a,n);
    for i:=1 to n do
      if prim(a[i]) then write(a[i],' ');
    readkey;
END.
```

❷ Se dă un vector cu $n (1 \le n \le 100)$ componente întregi de 1, 2, 3 sau 4 cifre. Să se afişeze cîte componente de fiecare fel conţine vectorul.

Rezolvare:

Vom folosi procedura **Cit_tablou** din exemplul precedent, funcţia **nc**(x), care returnează numărul de cifre ale numărului întreg x. Vom păstra rezultatul în tabloul $B(4)$, unde b_i reprezintă numărul componentelor de i cifre $(i = \overline{1, 4})$.

```
program Subpr2;
uses Crt;
type tablou=array [1..100] of integer;
var a:tablou;
    i,n,t:byte;
    b:[1..4] of byte;
procedure Cit_tablou(var t:tablou; var dim:byte);
var j:byte;
begin
   write('Dimensiunea vectorului: ');
   readln(dim);
   for j:=1 to dim do begin
      write(' Componenta ',j,': ');
      readln(t[j]);
   end;
end;
function nc(x:integer):byte;
var i,nr_cifre,intreg:integer;
begin
   i:=1; nr_cifre:=0;
   repeat
      intreg:=trunc(a/10*i));
      i:=i*10;
      inc(nr_cifre);
   until intreg=0;
   nc:=nr_cifre;
end;
BEGIN
   ClrScr;
   Cit_tablou(a,n);
```

```
   for i:=1 to n do begin
      t:=nc(a[i]);
      inc(b[t]);
   end;
   for i:=1 to 4 do
      writeln('Componente cu ',i,' cifre ',b[i]);
   readkey;
END.
```

❸ Se dă numărul natural n. Să se determine toate modalităţile în care pot fi plasate n regine pe o tablă de şah, astfel încît ele să nu se atace.

Rezolvare:

Pentru ca 2 regine să nu se atace, ele trebuie să nu se afle pe aceeaşi linie, coloană sau diagonală. Evident (deoarece sînt n regine), în fiecare linie este o regină. Vectorul $R[n]$, unde $R_i \in \{1, 2, ..., n\}$, conţine poziţia (numărul coloanei) ocupată de fiecare regină. Deci, R_i este numărul coloanei unde se află regina i. Prin urmare:

1. $R_i \neq R_j$, pentru orice $i, j \in \{1, ..., n\}$ $(i \neq j)$;

2. $|R_i - R_j| \neq |i - j|$, pentru orice $i, j \in \{1, ..., n\}$ $(i \neq j)$ (reginele i şi j nu se află pe aceeaşi diagonală).

```
program Subpr3;
uses Crt;
var r:array[1..20] of integer;
    n,j,nr_sol:integer;
    f:boolean;
function Verifica (j:integer):boolean;
var i:integer;
begin
   verifica:=true;
   for i:=1 to j-1 do
   if (r[i]=r[j]) or (abs(r[i]-r[j])=j-i) then begin
      verifica:=false;
      break;
   end;
end;
procedure solutie;
var i:integer;
begin
   inc(nr_sol);
   write('Solutia ',nr_sol,': ');
   for i:=1 to n do write(r[i],' ');
   writeln;
end;
BEGIN
   ClrScr;
   write('Introdu numarul de regine: ');
   readln(n);
   j:=1;
   nr_sol:=0;
   while j>0 do begin
      f:=false;
```

```
      while not f and (r[j]<=n-1) do begin
         inc(r[j]);
         if verifica(j) then f:=true;
      end;
      if not f then dec(j)
         else if j<n then begin inc(j); r[j]:=0; end
         else solutie;
   end;
   if nr_sol=0 then write('Nu exista solutii');
   readkey;
END.
```

Probleme propuse

A

1. Să se depisteze greşelile în următoarele declaraţii de funcţie:

   a)
   ```
   function M(x,y,z:integer):integer;
   begin
      if x>y then M:=x else M:=y;
      if z>M then M:=z;
   end;
   ```

   b)
   ```
   function S(x,y:integer):integer;
   var sum:integer;
   begin
      sum:=x+y;
   end;
   ```

   c)
   ```
   function f(x:real):real;
   begin
      f(x):=(exp(x)+exp(-x))/2;
   end;
   ```

2. Să se definească o funcţie pentru calcularea factorialului şi să se calculeze cu ajutorul ei combinări din n elemente luate cîte m. Numerele naturale m şi n sînt date. Formula de calcul: $C_n^m = \dfrac{n!}{m!(n-m)!}$.

3. Se dă un număr real x. Să se calculeze valoarea expresiei $S = \dfrac{\operatorname{ch} x + \operatorname{sh} x}{\operatorname{ch} x - \operatorname{sh} x}$, dacă $\operatorname{sh} x = \dfrac{e^x - e^{-x}}{2}$, $\operatorname{ch} x = \dfrac{e^x + e^{-x}}{2}$.

4. Fiind date numerele naturale a, b, c, să se determine cel mai mare divizor comun al acestor numere.

5. Definind funcţia-putere, să se calculeze valoarea expresiei:
$$S = 1 + 0{,}5^2 + 0{,}5^4 + 0{,}5^6 + 0{,}5^8.$$

6. Se dau numerele naturale a, b, c, d. Să se determine pentru fiecare 3 dintre cele patru numere, dacă ele pot fi lungimile laturilor unui triunghi. Dacă răspunsul este afirmativ, să se calculeze:
a) perimetrul triunghiului respectiv;
b) aria triunghiului respectiv.

7. Să se definească funcțiile $\max(a, b)$ și $\min(a, b)$, care returnează respectiv cel mai mare și cel mai mic dintre numerele reale a și b, apoi să se calculeze valoarea expresiei:
a) $S = \max(\min(a_1, a_2), \max(a_3, a_4)) + \min(\max(a_5, a_6), \min(a_7, a_8))$, unde $a_1, a_2, \ldots a_8$ sînt numere reale date;
b) $T = \min(a_1, a_2) + \min(a_3, a_4) + \ldots + \min(a_9, a_{10}) + \max(a_1, a_2) + \max(a_3, a_4) + \ldots + \max(a_9, a_{10})$, unde $a_1, a_2, \ldots, a_{10}$ sînt numere reale date.

8. Se dau numerele reale pozitive a, b, c, care sînt lungimile laturilor unui triunghi. Să se calculeze lungimile medianelor triunghiului.
Indicație. Lungimea medianei corespunzătoare laturii de lungimea a se calculează cu ajutorul formulei $m_a = 0{,}5\sqrt{2b^2 + 2c^2 - a^2}$.

9. Se dau numerele reale pozitive a, b, c, care sînt lungimile laturilor unui triunghi. Să se calculeze înălțimile triunghiului.
Indicație. Să se utilizeze formula $\mathcal{A} = \frac{h_a \cdot a}{2}$, unde $\mathcal{A}$ este aria tiunghiului, iar h_a – înălțimea corespunzătoare laturii a.

10. Se dă o mulțime de puncte în plan. Să se calculeze cea mai mică distanță dintre oricare 2 puncte posibile.

11. a) Să se descrie o funcție care va returna valoarea *true*, dacă numărul natural dat este prim, altfel – valoarea *false*.
b) Utilizînd funcția din a), să se afișeze toți divizorii primi ai numărului natural dat n.

12. a) Să se descrie o funcție care va returna numărul de divizori proprii ai numărului natural dat (divizorii proprii sînt diferiți de 1 și de numărul dat).
b) Utilizînd funcția din a), să se afișeze numerele naturale mai mici decît 10000, care au exact n divizori proprii, unde n este un număr natural dat mai mic decît 20.

13. Se dă un vector cu n ($1 < n < 100$) componente numere întregi. Să se afle:
a) cel mai mare divizor comun (*CMMDC*) al componentelor vectorului;
Indicație. $CMMDC(a_1, a_2, \ldots, a_n) = CMMDC(a_n, CMMDC(a_1, a_2, \ldots, a_{n-1}))$;
b) cel mai mic multiplu comun (*CMMMC*) al componentelor vectorului.
Indicație. $CMMMC(a_1, a_2, \ldots, a_n) = CMMMC(a_n, CMMMC(a_1, a_2, \ldots, a_{n-1}))$.

14. Să se definească o funcție care determină dacă un număr natural dat este pătrat perfect. Utilizînd această funcție, să se determine care dintre componentele unui vector dat de numere naturale sînt pătrate perfecte.

15. Să se definească un subprogram care va efectua:
a) adunarea a două fracții;
b) înmulțirea a două fracții;
c) simplificarea unei fracții pînă la o fracție ireductibilă;
d) compararea a două fracții.

16. Să se definească un subprogram care va:
a) aduna două numere complexe;
b) înmulți două numere complexe;
c) împărți două numere complexe;
d) calcula modulul unui număr complex;
e) ridica la putere un număr complex.

17. Să se definească un subprogram care va:
a) aduna două matrice pătrate;
b) înmulți două matrice pătrate;
c) calcula transpusa unei matrice pătrate;
d) calcula inversa unei matrice pătrate.
Să se rezolve ecuația $AX = B$, unde A și B sînt matrice pătrate date.

18. Să se definească un subprogram care va:
a) aduna doi vectori din plan (fiind date coordonatele lor);
b) calcula produsul scalar a doi vectori din plan;
c) înmulți un vector din plan cu un scalar;
d) calcula lungimea unui vector din plan.

19. Să se definească un subprogram care va:
a) aduna măsurile a două unghiuri (exprimate în grade, minute, secunde);
b) scădea măsurile a două unghiuri;
c) înmulți măsura unui unghi cu un număr natural;
d) împărți cu rest măsurile a două unghiuri;
e) compara măsurile a două unghiuri.

20. Se dau coordonatele a n ($3 < n < 50$) puncte din plan. Definind o funcție care determină dacă 3 puncte sînt sau nu coliniare, să se determine dacă fiecare 3 din cele n puncte sînt necoliniare.

21. Să se definească un subprogram care va:
a) aduna două intervale de timp (exprimate în secunde, minute, ore, zile, săptămini);
b) scădea două intervale de timp;
c) înmulți un interval de timp cu un număr natural;
d) împărți cu rest două intervale de timp;
e) compara două intervale de timp.

22. Să se definească un subprogram care va:
a) aduna o dată calendaristică (o dată calendaristică este exprimată de un triplet de forma (zi, luna, an)) cu un număr de zile, rezultatul fiind o dată calendaristică;

b) aduna o dată calendaristică cu un număr de luni, rezultatul fiind o dată calendaristică;
c) scădea dintr-o dată calendaristică un număr de zile, rezultatul fiind o dată calendaristică;
d) scădea dintr-o dată calendaristică un număr de luni, rezultatul fiind o dată calendaristică;
e) scădea două date calendaristice, rezultatul fiind numărul de zile dintre aceste date.

B

23. a) Să se descrie o funcţie care va efectua adunarea a două numere naturale, al cărei rezultat poate avea pînă la 255 de cifre.
b) Utilizînd funcţia din a), să se realizeze un program pentru înmulţirea unui număr natural cu un număr mai mic decît 100, al cărei rezultat poate avea pînă la 255 de cifre.

24. Să se descrie o funcţie care va efectua înmulţirea a două numere naturale, fiecare conţinînd pînă la 100 de cifre.

25. Să se descrie o funcţie care va efectua împărţirea (exactă sau cu rest) unui număr ce conţine pînă la 255 de cifre la un număr mai mic decît 1 000.

26. Să se efectueze adunarea, scăderea şi înmulţirea a două numere date într-o bază dată, diferită de 10. Rezultatul se va afişa în aceeaşi bază.
Indicaţie. Se pot utiliza două funcţii: una pentru conversia în sistemul zecimal, alta – din sistemul zecimal.

27. *Problema lui Fermat*[1]. Să se găsească un număr, astfel încît suma cubului numărului şi divizorilor proprii ai numărului să fie pătrat perfect.

28. Se dă numărul natural n. Să se determine al n-lea termen din şirul lui Fibonacci:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

De exemplu, pentru $n = 8$ se va afişa 21.

29. Se dau numerele naturale b şi n, unde $1 < b < 10$. Să se definească un subprogram care va:
a) verifica dacă un număr dat este scris corect în sistemul de numeraţie cu baza b;
b) aduna două numere scrise în sistemul de numeraţie cu baza b;
c) scădea două numere scrise în sistemul de numeraţie cu baza b;
d) înmulţi două numere scrise în sistemul de numeraţie cu baza b;
e) împărţi două numere scrise în sistemul de numeraţie cu baza b;
f) transforma un număr din sistemul de numeraţie cu baza b în sistemul zecimal de numeraţie;
g) transforma un număr din sistemul zecimal de numeraţie în sistemul de numeraţie cu baza b.

[1] Pierre Fermat (1601–1665) – matematician francez.

30. Să se definească un subprogram care va calcula derivata unui polinom de o nedeterminată. Pentru reprezentarea polinomului să se utilizeze tipul `string`. De exemplu, `'3*x^4-12*x^3+125*x-39'` reprezintă polinomul $3X^4 - 12X^3 + 125X - 39$.

31. *Ipotezele lui Goldbach*[1].

a) Să se reprezinte fiecare număr par mai mare sau egal cu 6 şi mai mic decît 1 000 ca sumă a două numere prime impare. De exemplu,
$100 = 97 + 3 = 89 + 11 = 71 + 29 = ...$

b) Se se reprezinte fiecare număr natural mai mare decît 1 şi mai mic decît 1 000 ca sumă a cel mult trei numere prime.

32. Să se descrie o funcţie care va efectua împărţirea cu rest a două numere naturale, fiecare fiind compus din cel mult 255 de cifre.

33. *Formulele lui Ramanujan*[2].

Să se calculeze:

a) $\sqrt{1+2\sqrt{1+3\sqrt{1+4\sqrt{1+...}}}}$

Să se compare rezultatul cu 3.

b) $\sqrt{8-\sqrt{8+\sqrt{8-\sqrt{8-\sqrt{8+\sqrt{8-...}}}}}}$, unde semnele rădăcinilor se repetă în grup cîte trei: –, +, –.

Să se compare rezultatul cu $1+2\sqrt{3}\sin 20°$.

c) $\sqrt{11-2\sqrt{11+2\sqrt{11-2\sqrt{11-2\sqrt{11+...}}}}}$, unde semnele rădăcinilor se repetă în grup cîte trei: –, +, –.

Să se compare rezultatul cu $1+4\sin 10°$.

d) $\sqrt{23-2\sqrt{23+2\sqrt{23+2\sqrt{23-2\sqrt{23+...}}}}}$, unde semnele rădăcinilor se repetă în grup cîte trei: –, +, +.

Să se compare rezultatul cu $1+4\sqrt{3}\sin 20°$.

---

[1] Christian Goldbach (1690–1764) – matematician rus.

[2] Scrinivasa Ramanujan (1887–1920) – matematician indian.

12 Subprograme recursive

Sugestii teoretice

Un subprogram se numeşte **subprogram recursiv** dacă el conţine apeluri la el însuşi.

Un subprogram recursiv se numeşte **subprogram direct recursiv** dacă în corpul lui apar apeluri la el însuşi.

Dacă un subprogram S_1 conţine apeluri la un subprogram S_2, iar S_2 face apeluri la S_1, atunci fiecare dintre subprogramele S_1 şi S_2 se numeşte **subprogram indirect recursiv**.

Se spune că S_1 şi S_2 sînt **subprograme mutual recursive**.

Pentru organizarea recursivităţii indirecte antetul unuia dintre cele două subprograme mutual recursive se declară anticipat. El este urmat de directiva **forward**. Ulterior, mai jos în program antetul se rescrie (fără cuvîntul `forward` şi fără lista parametrilor formali) împreună cu corpul subprogramului.

Probleme rezolvate

❶ Se dă un vector cu n, $n < 100$, componente întregi. Utilizînd o funcţie recursivă, să se determine componenta minimală.

Rezolvare:

```
program Recur1;
uses Crt;
var a:array[1..100] of integer;
    i,n:integer;
function Min(x,y:integer):integer;
begin
   if x>y then Min:=y else Min:=x;
end;
function min_mas(n:integer):integer;
begin
   if n=2 then min_mas:=Min(a[1],a[2]) else
      min_mas:=Min(a[n],min_mas(n-1));
end;
BEGIN
   ClrScr;
   write('Introdu numarul de elemente: ');
   readln(n);
   for i:=1 to n do begin
      write('A[',i,']=');
```

```
      readln(a[i]);
   end;
   writeln('Min= ',min_mas(n));
   readkey;
END.
```

❷ Să se elaboreze un algoritm pentru calcularea c.m.m.d.c. (*CMMDC*) al numerelor întregi date m şi n utilizînd algoritmul lui Euclid.

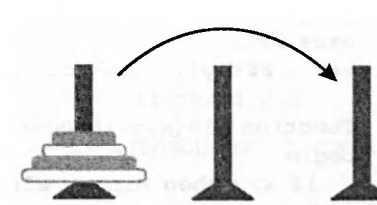
Rezolvare:

Conform algoritmului lui Euclid (pentru $m > n$):

$$CMMDC(m,n) = \begin{cases} n, \text{ dacă } n \text{ divide } m \\ CMMDC(n,r), \text{ unde } r \text{ este restul împărţirii lui } m \text{ la } n, \text{ dacă } n \text{ nu divide } m. \end{cases}$$

```
program Recur2;
uses Crt;
var m,n:integer;
function CMMDC(a,b:integer):integer;
begin
   if a mod b=0 then CMMDC:=b
      else CMMDC:=CMMDC(b,a mod b);
end;
BEGIN
   ClrScr;
   write('Introdu 2 numere intregi: ');
   readln(m,n);
   write('CMMDC(',m,',',n,')=');
   if m>n then write(CMMDC(m,n))
      else write(CMMDC(n,m));
   readkey;
END.
```

❸ *Turnurile din Hanoi*



Cîndva demult în oraşul vietnamez Hanoi erau trei turnuri. Pe primul turn erau îmbrăcate n discuri de diferite dimensiuni în ordinea descrescătoare a dimensiunilor (cel mai mare disc era dedesubt, iar cel mai mic – deasupra). Se cere să se mute cele n discuri pe turnul al treilea respectînd regulile:

- la fiecare mişcare se mută un singur disc;
- un disc nu poate fi plasat peste unul mai mic;
- turnul din mijloc poate fi folosit ca poziţie intermediară.

Rezolvare:

Considerăm că discurile sînt notate prin $1, 2, \ldots, n$ în ordinea creşterii dimensiunilor lor. Fie că ştim a muta primele $n-1$ discuri. În asemenea situaţie, pentru a muta toate cele n discuri, vom proceda astfel:

Pasul 1. Mutăm primele $n-1$ discuri de pe primul turn pe al 2-lea, folosindu-l pe al 3-lea.

Pasul 2. Mutăm discul al n-lea de pe primul turn pe al 3-lea.

Pasul 3. Mutăm cele $n - 1$ discuri de pe al 2-lea turn pe al 3-lea, folosindu-l pe primul.

```
program Recur3;
uses Crt;
var n,i:integer;
procedure Muta(n:integer; X,Y,Z:char); {X este primul turn,
   Y este turnul al 2-lea, iar Z este turnul al 3-lea}
begin
   if n>=1 then begin
     muta(n-1,X,Z,Y);{pasul 1}
     write(X,'->',Z,'   ');{pasul 2}
     inc(i);
     if i mod 8=0 then writeln;{afiseaza cite 8 mutari in rind}
     muta(n-1,Y,X,Z);{pasul 3}
   end;
end;
BEGIN
   ClrScr;
   write('Input number of rings: ');
   readln(n);
   muta(n,'X','Y','Z');
   readkey
END.
```

❹ Se dau numerele naturale $g_1, g_2, \ldots, g_n$ reprezentînd n greutăți și numărul natural m reprezentînd o altă greutate (masa). Să se determine dacă se poate forma din greutățile date (nu neapărat toate) masa m. De exemplu, pentru șirul de greutăți 10, 2, 4, 7, 3, 2, 5, 10 și $m = 11$ una dintre soluții este 2, 3, 4, 2 (2 + 3 + 4 + 2 = 11).

Rezolvare:

Evident, dacă greutatea g_1 este favorabilă (va fi inclusă în soluție), atunci din greutățile rămase trebuie să putem forma masa $m - g_1$. Același raționament este valabil și pentru g_2 ș.a.m.d. De exemplu, greutatea 10 din șirul din enunț nu este favorabilă, deoarece din restul greutăților nu se poate forma masa 1 = 11 – 10.

Funcția **Favorabil**(m, i):`boolean` returnează valoarea *true*, dacă putem forma din greutățile $g_i, g_{i+1}, \ldots, g_n$ masa m. Deci, g_i este favorabilă dacă și numai dacă **Favorabil**($m - g_i$, $i + 1$) returnează *true*.

```
program Recur4;
uses Crt;
var n,i,m:integer;
    g:array[1..50] of integer;
function Favorabil(m,i:integer):boolean;
begin
   if m=0 then Favorabil:=true else
      if (m<0) or (i>n) then Favorabil:=false else
         if Favorabil(m-g[i],i+1) then begin
            write(g[i],' ');
            Favorabil:=true;
         end
         else Favorabil:=Favorabil(m,i+1);
end;
```

```
BEGIN
   ClrScr;
   write('Introdu numarul de greutati: ');
   readln(n);
   for i:=1 to n do begin
      write('Greutatea ', i,': ');
      readln(g[i]);
   end;
   write('Introdu masa: ');
   readln(m);
   if not Favorabil(m,1) then write('Problema n-are solutii!');
   readkey;
END.
```

❺ Se dă numărul natural n. Să se afişeze toate descompunerile posibile ale numărului n. De exemplu, pentru $n = 3$ se va afişa:

3

1 + 2

2 + 1

1 + 1 + 1

Rezolvare:

```
program Recur5;
uses Crt;
var n,i,s,j:integer;
    a:array[1..100] of integer;
procedure Desc(j,s,k:integer);{descompune numarul s ca suma de k numere}
var i:integer;
begin
   if (s=0) and (j=k+1) then begin
      for i:=1 to j-1 do write(a[i],'+');
      writeln(#8,' '); {ultimul + va fi substituit cu un spatiu}
   end else for i:=1 to s do begin
      a[j]:=i;
      Desc(j+1, s-i,k);
   end;
end;
BEGIN
   ClrScr;
   write('Scrie numarul: ');
   readln(n);
   for i:=1 to n do
      Desc(1,n,i);
   readkey;
END.
```

❻ Se dau numerele naturale n şi k. Să se genereze toate combinaţiile de lungime n (din n numere), formate din numere din mulţimea $\{1, ..., k\}$. De exemplu, pentru $n = 3$ şi $k = 2$ se va afişa:

1 1 1
1 1 2
1 2 1
1 2 2
2 1 1
2 1 2
2 2 1
2 2 2

Rezolvare:

```
program Recur6;
uses Crt;
var a:array[1..30] of integer;
    n,k,t:integer;
procedure genereaza;
var i,j:integer;
begin
   if t=n then begin
      for i:=1 to n do write(a[i],' ');
      writeln;
   end else
      for j:=1 to k do begin
         t:=t+1;
         a[t]:=j;
         genereaza;
         t:=t-1;
      end;
end;
BEGIN
   ClrScr;
   write('Introdu n,k: ');
   readln(n,k);
   t:=0;
   genereaza;
   readkey;
END.
```

❼ Să se elaboreze un algoritm care deplasează linia ---------- de sus în jos pînă aceasta atinge ultimul rind al ecranului, apoi o deplasează în sus pînă atinge primul rînd, după care o deplasează iar în jos ş.a.m.d.

```
program Recur7; {Recursivitate indirecta}
uses Crt;
procedure sus; forward;{declaratie anticipata de procedura}
procedure jos;
var i:byte;
begin
   i:=0;
   repeat
      inc(i);
      GotoXY(30,i);
```

```
      write('-----------');
      delay(300);
      delline;{sterge linia}
      if keypressed then exit;
      {daca se va apasa o tasta se va iesi din procedura}
   until i=25;
   Sus;
end;
procedure Sus;
var i:byte;
begin
   i:=24;
   Delline;
   repeat
      dec(i);
      GotoXY(30,i);
      write('-----------');
      delay(300);
      ClrScr;
      if keypressed then exit;
   until i=1;
   Jos;
end;
BEGIN
   Sus;
END.
```

⑧ Fie n localități notate cu numerele 1, 2, ..., n. Între fiecare două dintre aceste localități există sau nu drum. Fiind date două localități să se determine dacă se poate ajunge dintr-o localitate în alta pe drumurile date. Informația despre drumuri se citește sub formă de consecutivitate de perechi de forma $[i,j]$, unde $i<j$ și $i, j \leq n$, semnificînd faptul că între localitățile i și j există drum.

Rezolvare:

```
program Recur8;
uses Crt;
var  graf:array[1..10,1..10] of 0..1; {matricea drumurilor}
     drum:array[1..10] of integer;
     vizitat:array[1..10] of boolean;
     start,final,i,j,n,lg,x,y:integer;    {n este numarul de orase, lg -
                                           num. de legaturi}
     stop:boolean;
procedure pas(s,f,m:integer); {al m-lea oras}
var i,j:integer;
begin
   if s=f then begin
      stop:=true;
      writeln('Calea este: ');
      for i:=1 to m-1 do write(drum[i],' ');
      writeln;
   end
```

```
    else begin
      for j:=1 to n do begin
        if (graf[s,j]<>0) and (not vizitat[j]) then begin
          drum[m]:=j;
          vizitat[j]:=true;
          pas(j,f,m+1);
          vizitat[j]:=false;
          drum[m]:=0;
        end;
      end;
    end;
  end;
BEGIN
  ClrScr;
  write('Introdu numarul de orase: ');
  readln(n);
  for i:=1 to n do
    for j:=1 to n do graf[i,j]:=0;
  for i:=1 to n do vizitat[i]:=false;
  write('Introdu numarul de legaturi directe: ');
  readln(lg);
  for i:=1 to lg do begin
    write('Legatura ',i,': ');
    readln(x,y);
    graf[x,y]:=1;
    graf[y,x]:=1;
  end;
  write('Introdu orasul de pornire: ');
  readln(start);
  drum[1]:=start;
  vizitat[start]:=true;
  write('Introdu orasul de sosire: ');
  readln(final);
  pas(start, final, 2);
  if not stop then
    writeln('Nu se poate ajunge din orasul ',start,' in orasul ',
      final);
  readkey;
END.
```

Probleme propuse

A

1. Se dau numerele naturale m şi n. Să se calculeze $CMMDC(m, n)$, dacă:
   - pentru $m > n$, $CMMDC(m, n) = CMMDC(m - n, n)$;
   - pentru $m = n$, $CMMDC(m, n) = m$.

2. Şirul lui Fibonacci se defineşte astfel:
   $F_1 = F_2 = 1$, $F_n = F_{n-1} + F_{n-2}$ pentru $n > 2$.
   Să se determine al n-lea termen din şirul lui Fibonacci.

3. Să se calculeze coeficienții binomiali C_1^n, C_n^2, ..., C_n^n obținuți în urma dezvoltării unui binom la puterea n ($n > 2$), știind că are loc următoarea relație de recurență:

$C_n^k = \frac{n-k+1}{k} C_n^{k-1}$ și $C_0^n = 1$, pentru orice $1 \le k \le n$.

4. Se dă numărul natural n. Să se calculeze suma:

a) $2 \cdot 2 + 3 \cdot 4 + 4 \cdot 6 + 5 \cdot 8 + 6 \cdot 10 + ... + n \cdot 2(n-1)$;

b) $S_n = \frac{1}{(1 \cdot 3)} + \frac{2}{(3 \cdot 5)} + \frac{3}{(5 \cdot 7)} + ... + \frac{n}{(2n-1) \cdot (2n+1)}$.

5. Se citește de la tastatură numărul natural n. Folosind un subprogram recursiv, să se calculeze suma $1 + 2 + ... + n$.

6. Să se calculeze valoarea funcției $f: \mathbb{N} \to \mathbb{N}$, $f = \begin{cases} n-1, \text{ dacă } n > 10 \\ f(f(n+2)), \text{ dacă, } n \le 10 \end{cases}$ pentru argumentul dat n.

B

7. Să se formuleze problema care se rezolvă cu ajutorul următorului algoritm:

a)
```
program Recur1;
uses Crt;
var  n:byte;
   r:integer;
function s(m:byte):integer;
var a:byte;
begin
   readln(a);
   if m=1 then s:=a else s:=a+s(m-1);
end;
BEGIN
   ClrScr;
   readln(n);
   r:=s(n);
   write('R=',r);
   readkey;
END.
```

b)
```
program Recur2;
uses Crt;
var  a,n:byte;
   r:longint;
function p(a,n:byte):longint;
begin
   if n=1 then p:=a else p:=a*p(a,n-1);
end;
BEGIN
   ClrScr;
   readln(a,n);
   r:=p(a,n);
```

```
    write('R=',r);
    readkey;
END.
```

c)
```
program Recur3;
uses Crt;
procedure Scrie;
var car:char;
begin
    read(car);
    if car<>#13 {codul tastei Enter} then Scrie;
    write(c);
end;
BEGIN
    ClrScr;
    write('Scrie: ');
    Scrie;
    readkey;
END.
```

8. Se dă numărul natural n, $n > 1$, şi numărul real x. Să se calculeze:
a) valoarea polinoamelor Hermite[1] $H_i(x)$, $i = 1, ..., n$, definite prin relaţiile:
$H_0(x) = 1$, $H_1(x) = 2x$, $H_i(x) = 2xH_{i-1}(x) - 2(i-1)H_{i-2}(x)$, unde $i = 2, ..., n$;
b) valoarea polinoamelor Legendre[2] $L_i(x)$, $i = 1, ..., n$, definite prin relaţiile:
$L_0(x) = 1$, $L_1(x) = x$, $L_i(x) = \frac{1}{i}[(2i-1)xL_{i-1}(x) - (i-1)L_{i-2}(x)]$, unde $i = 2, ..., n$;
c) valoarea polinoamelor ortogonale Cebîşev[3] $C_i(x)$, $i = 1, ..., n$ şi $|x| \le 1$, definite prin relaţiile: $C_0(x) = 1$, $C_1(x) = x$, $C_i(x) = 2iC_{i-1}(x) - C_{i-2}(x)$, unde $i = 2, ..., n$. Să se compare cu rezultatele obţinute prin formula $C_i(x) = \cos(i \cdot \arccos x)$.

9. Se dă numărul natural n, $n < 65\,000$, şi numărul natural b de o cifră. Să se scrie reprezentarea numărului n în baza b.

10. Se dă numărul natural n. Folosind un subprogram recursiv, să se afişeze răsturnatul lui n. De exemplu, pentru $n = 3\,492$ se va afişa $2\,943$.

11. Se dă numărul natural n. Să se calculeze valorile funcţiilor $f, g: \mathbb{N} \to \mathbb{N}$, în n, dacă
$f(0) = 1, f(1) = 2$,
$g(0) = 3, g(1) = 4$,
$f(x) = f(x-1) + g(x-1)$,
$g(x) = 2g(x-1) + 3f(x-1)$, pentru orice număr natural $x > 1$.

---

[1] Andrien Marie Legendre (1752–1833) – matematician francez.

[2] Charles Hermite (1822–1901) – matematician francez.

[3] Pafnuti Lvovici Cebîşev (1821–1894) – matematician rus.

12. Se dă un fişier *text* cu numere naturale, după care urmează un număr întreg negativ. Folosind un subprogram recursiv, să se calculeze suma numerelor naturale din fişier.

13. Se dă numărul natural n. Să se genereze toate submulţimile mulţimii $\{1, 2, ..., n\}$.

C

14. Se dă o matrice pătrată $A(n, n)$ de numere întregi. Să se calculeze determinantul matricei, utilizînd formula descompunerii după prima linie:

$$\det A = \sum_{i=1}^{n} (-1)^{i+1} a_{i1} \cdot \det B_i,$$

unde B_i este matricea obţinută din A eliminînd prima linie şi coloana i.

15. Se dă numărul natural n. O persoană trebuie să urce o scară cu n trepte. Ştiind că la fiecare pas persoana poate să urce una sau două trepte (evident, consecutive), să se determine numărul de moduri în care persoana poate urca scara.

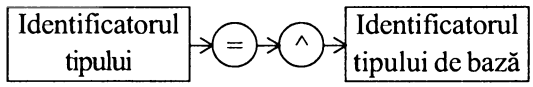
16. Fie n localităţi notate cu numerele $1, 2, ..., n$. Între fiecare două dintre aceste localităţi există sau nu drum. În cazul în care există, se ştie şi lungimea drumului. Fiind date două localităţi, să se determine dacă se poate ajunge dintr-o localitate în alta şi să se determine drumul de lungime minimă dintre aceste localităţi.

13 Alocarea dinamică a memoriei (Tipul *referință*). Structuri dinamice de date

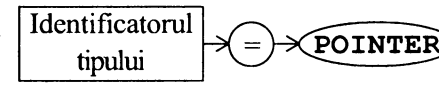
Sugestii teoretice

Declararea tipului *referință*

Indicatori tipizați



Indicatori netipizați



Observații

1. Tipul de bază poate fi declarat înainte sau după denumirea tipului *referință*.

2. Valorile variabilelor-referință sînt adrese. Ele nu pot fi citite de la tastatură și afișate pe ecran.

3. Adresa unei variabile *v* se determină cu ajutorul funcției addr (*v*) sau cu ajutorul operatorului @.

4. Valoarea unei variabile-referință poate fi NIL (numită pointer vid) și nu indică nici o adresă.

Exemple:

```
type tp=^name;
     name=record
        nume,prenume:string[20];
     end;
var p:tp; {variabila-referinta la tipul name}
    t:^integer; {variabila-referinta la tipul integer}
    q:^real; {variabila-referinta la tipul real}
    r:pointer; {variabila-referinta netipizată}
    i:real;
    j:integer;
BEGIN
   ...
   t:=NIL;
   q:=@i; {valoarea variabilei q este adresa variabilei statice i}
   t:=addr(j); {valoarea variabilei t este adresa variabilei statice j}
   r:=@i;
   r:=@j;
   ...
```

Observaţie

Asupra variabilelor-referinţă (de acelaşi tip de bază) pot fi aplicaţi operatorii := (atribuirea), <> (neegalitatea), = (egalitatea).

Variabile dinamice

Variabilele dinamice sînt variabile create şi distruse în timpul execuţiei programului. Ele sînt generate de variabilele-referinţă, care memorizează adresele zonelor de memorie alocate variabilelor dinamice.

Crearea variabilelor dinamice

New(p), unde p este o variabilă-referinţă tipizată, – creează variabila dinamică p^ (alocă un spaţiu de memorie egal cu numărul de octeţi necesari păstrării unei valori a tipului de bază şi returnează adresa spaţiului prin variabila p).

GetMem(p, size), unde p este de tip pointer, iar size de tip word, – creează variabila dinamică p^ (alocă un spaţiu de memorie de Size octeţi şi returnează adresa spaţiului prin variabila p).

Distrugerea variabilelor dinamice

Dispose(p), unde p este o variabilă-referinţă tipizată, – distruge variabila dinamică p^ (eliberează spaţiul de memorie alocat anterior lui p^ prin procedura New, după care valoarea lui p devine nedefinită).

FreeMem(p), unde p este de tip pointer, – distruge variabila dinamică p^ (eliberează spaţiul de memorie alocat anterior lui p^ prin procedura GetMem, după care valoarea lui p devine nedefinită).

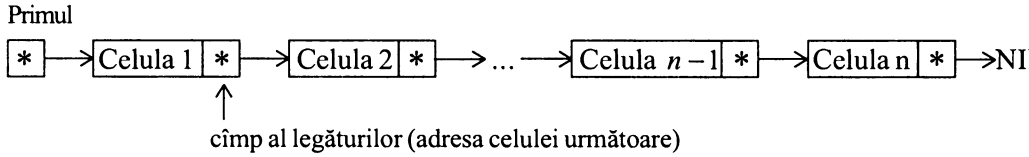
Liste înlănţuite, stive, cozi

Listele înlănţuite sînt structuri de date explicite, dinamice, omogene, cu acces secvenţional, formate din **celule**.

Fiecare **celulă** este o variabilă dinamică avînd tipul de bază record, care în afară de cîmpurile datelor conţine un **cîmp al legăturilor** (sau două) – cîmp ce conţine adresa celulei la care se poate ajunge din celula curentă.

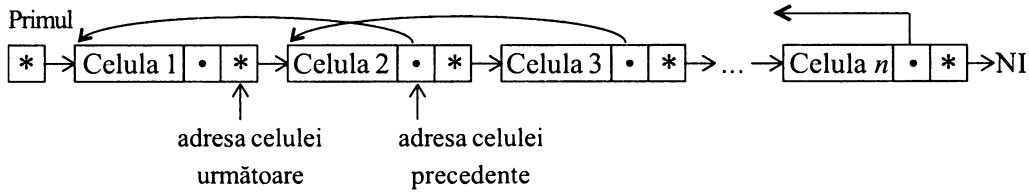
În funcţie de numărul de „vecini", a căror adresă se păstrează în fiecare celulă, şi de forma generală a structurii, există următoarele tipuri de liste:

a) liste simplu înlănţuite (unidirecţionale)



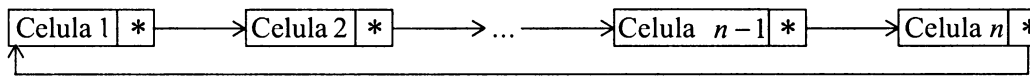
```
type  pcelula=^celula;
      celula=record
         cimpuri cu date;
         urmator:pcelula;
      end;
```

b) **liste dublu înlănțuite (bidirecționale)**

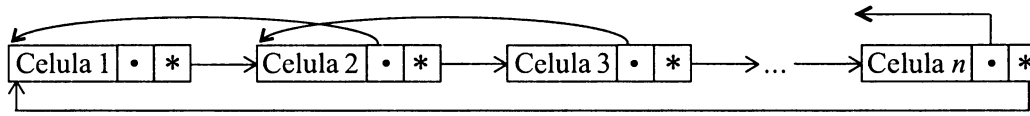



```
type  pcelula=^celula;
      celula=record
         cimpuri cu date;
         anterior, urmator:pcelula;
      end;
```

c) **liste circulare simplu înlănțuite**



d) **liste circulare dublu înlănțuite**



Stiva este o structură de date asupra cărora se pot efectua două operații:

a) introducerea unui element în stivă, deasupra celor existente (dacă există);

b) extragerea elementului din vîrful stivei, dacă acesta există.

Stiva este o structură de date de tip LIFO (Last In First Out – ultimul intrat, primul ieşit), adică ordinea de extragere a elementelor din stivă este inversă celei în care acestea au fost introduse.

În continuare, pentru organizarea unei stive, vom utiliza o listă unidirecțională cu proprietatea că operațiile de introducere şi extragere se vor efectua la un singur capăt al listei. De asemenea, putem organiza o stivă, utilizînd un vector (pentru păstrarea elementelor) şi o variabilă de tip întreg (care va fi vîrful stivei, deci va păstra poziția ocupată de ultimul element introdus în stivă).

Coada este o structură de date asupra cărora se pot efectua două operații:

a) introducerea unui element în coadă, după cele existente (dacă există);

b) extragerea elementului din vîrful cozii, dacă acesta există.

În continuare, pentru organizarea unei cozi, vom utiliza o listă unidirecțională cu proprietatea că operația de introducere se efectuează la unul dintre capete, iar cea de extragere – la celălat capăt al listei.

Probleme rezolvate

❶ a)Să se creeze un algoritm care citeşte de la tastatură informaţii despre un grup de persoane (numele şi vîrsta fiecărei persoane) şi crează o listă. Pentru a finisa lista se va scrie stringul 'STOP'.

b) Să se afişeze lista la ecran.

c) Să se includă în lista după celula ce conţine informaţii despre persoana Albu (se presupune că această celulă există) informaţii despre o persoană nouă.

d) Să se excludă din listă celula ce conţine informaţii despre persoana Albu.

Rezolvare:

```
program lista;
Uses Crt;
type legatura=^celula;
     celula=record
        name:string[20];
        age:byte;
        next:legatura;
     end;
var primul,p,q,nou: legatura;
    nume: string[20];
    n: integer;
BEGIN
ClrScr;

{ A. Crearea listei }
writeln('Pentru a finisa introducerea datelor scrie STOP');
primul:=nil;
write('Numele persoanei: ');
readln(nume);
while nume<>'STOP'do begin
   new(p);
   p^.name:=nume;
   write('Virsta persoanei: ');
   readln(p^.age);
   p^.next:=primul;
   primul:=p;
   write('Numele persoanei: ');
   readln(nume);
end;

{ B. Afisatea listei}
writeln('---------- Lista persoanelor ----------');
p:=primul;
while p<>nil do begin
     writeln(p^.name:20, '  ', p^.age:4);
     p:=p^.next;
     inc(n);
     end;
     writeln('In total, in lista sint ', n, ' persoane');
readkey;
```

```
{ C. Includerea in lista a unei noi peroane dupa persoana Albu}
p:=primul;
new(nou);
write('Numele persoanei noi: ');
readln(nou^.name);
write('Virsta persoanei noi: ');
readln(nou^.age);
while p^.name<>'Albu' do
   p:=p^.next;
nou^.next:=p^.next; {din celulele Nou si P se poate ajunge
                     la aceeasi celula}
p^.next:=nou;
{Afisatea listei dupa includere}
writeln('----- Lista noua a persoanelor (dupa includere) -------');
p:=primul;
while p<>nil do begin
   writeln(p^.name:20, '  ', p^.age:4);
   p:=p^.next;
end;
readkey;

{ D. Excluderea din lista a persoanei Albu }
q:=primul; {q este celula precedenta celulei cu Albu}
while q^.next^.name<>'Albu' do
   q:=q^.next;
p:=primul; {p este celula cu Albu}
while p^.name<>'Albu' do
   p:=p^.next;
q^.next:=p^.next;
dispose(p);
{Afisatea listei dupa excluderea lui Albu}
writeln('----- Lista noua a persoanelor (dupa excluderea lui
        Albu) -------');
p:=primul;
while p<>nil do begin
   writeln(p^.name:20, '  ', p^.age:4);
   p:=p^.next;
end;
readkey;
END.
```

❷ a) Să se creeze un algoritm care citeşte de la tastatură informaţii despre un grup de persoane (numele şi vîrsta fiecăreia) şi creează o listă în ordine alfabetică (lexicografică) a numelor.

b) Să se calculeze numărul de persoane cu vîrsta mai mică de 20 de ani.

c) Să se insereze în listă cîteva persoane.

d) Să se şteargă o persoană din listă (fiind dat numele ei).

e) Să se păstreze lista într-un fişier-text.

Rezolvare:
Vom crea o listă simplu înlănțuită.

```
program lista;
uses Crt;
type legatura=^persoana;
     persoana=record
        name:string[20];
        age:byte;
        next:legatura
     end;
var  primul:legatura;
     nume:string[20];
     n:integer;
     c:char;
function nr_pers20(First:legatura):integer;
var  contor:integer;
     ind_curent:legatura;
begin
   contor:=0;
   ind_curent:=first;
   while ind_curent<>nil do begin
      if ind_curent^.age<20 then inc(contor);
      ind_curent:=ind_curent^.next;
   end;
   nr_pers20:=contor;
end;
procedure afisare_lista(first:legatura);
var  ind_curent:legatura;
begin
   ind_curent:=first;
   while ind_curent<>nil do begin
      writeln(ind_curent^.name,' ',ind_curent^.age);
      ind_curent:=ind_curent^.next;
   end;
end;
procedure sterge(var first:legatura; nume:string);
var  ind_curent,temp:legatura;
begin
   ind_curent:=first;
   if first^.name=nume then begin
      first:=first^.next;
      dispose(ind_curent);
      exit;
   end;
   while (ind_curent^.next^.name<>nume) and (ind_curent^.next<>nil) do
      ind_curent:=ind_curent^.next;
   if ind_curent^.next=nil then exit;
   temp:=ind_curent^.next;
   ind_curent^.next:=ind_curent^.next^.next;
   dispose(temp);
end;
procedure in_fisier_lista(first:legatura);
var  ind_curent:legatura;
     f:text;
```

```
begin
   assign(f,'c:\lista.txt');
   rewrite(f);
   ind_curent:=first;
   while ind_curent<>nil do begin
      writeln(f,Ind_curent^.name,'  ',ind_curent^.age);
      Ind_curent:=Ind_curent^.next;
   end;
   close(f);
end;
procedure lista_ord(var first:legatura;m:integer);
var  j:integer;
     q,temp:legatura;
begin
   new(first);
   write('Numele persoanei 1: ');readln(first^.name);
   write('Virsta persoanei 1: ');readln(First^.age);
   first^.name[1]:=upcase(first^.name[1]);
   first^.next:=nil;
   for j:=2 to m do begin
      new(q);
      write('Numele persoanei ', j,': ');readln(q^.name);
      write('Virsta persoanei ', j,': ');readln(q^.age);
      q^.name[1]:=upcase(q^.name[1]);
      if q^.name<=First^.name then begin
         q^.next:=first;
         first:=q;
      end;
      else begin
         temp:=first;
         while (temp^.next<>nil) and (q^.name>temp^.next^.name) do
            temp:=temp^.next;
         q^.next:=temp^.next;
         temp^.next:=q;
      end;
   end;
end;
procedure ins_pers(var first:legatura);
var q,temp:legatura;
begin
   new(q);
   write('Numele persoanei: ');readln(q^.name);
   write('Virsta persoanei: ');readln(q^.age);
   q^.name[1]:=upcase(q^.name[1]);
   if q^.name<=first^.name then begin
     q^.next:=first;
     first:=q;
   end;
   else begin
     temp:=first;
     while (temp^.next<>nil) and (q^.name>temp^.next^.name) do
        temp:=temp^.next;
```

```
      q^.next:=temp^.next;
      temp^.next:=q;
    end;
end;
BEGIN
   ClrScr;
   writeln('---- Formarea listei --------'); {a}
   write('Introdu numarul de persoane: '); readln(n);
   lista_ord(primul,n);
   writeln('Numarul de persoane cu virsta mai mica de 20 ani: ',
     nr_pers20(primul)); {b}
   afisare_lista(primul);
   repeat {c}
     write('Mai scriem o persoana? (y/n): ');readln(c);
     if c='y' then ins_pers(primul);
   until c<>'y';
   afisare_lista(primul);
   write('Scrie numele persoanei ce urmeaza a fi eliminata: ');{d}
   readln(nume);
   sterge(primul,nume);
   writeln('========================');
   afisare_lista(primul);
   in_fisier_lista(primul);{e}
   readkey;
END.
```

❸ Să se creeze şi să se afişeze o listă dublu înlănţuită (sau bidirecţională) ale cărei componente vor fi string-uri.

Rezolvare:

```
program Lista_bi; {Lista bidirectionala}
uses Crt;
type legatura=^Comp;
     Comp=record
        s:string;
        anti,next:legatura;
     end;
var primul:legatura;
    n:integer;
procedure Creeaza_lista(var p:legatura);
var j:integer;
    temp:legatura;
begin
   new(p);
   write('Introdu primul: ');
   readln(p^.s);
   p^.anti:=NIL;
   p^.next:=NIL;
   for j:=2 to n do begin
      new(temp);
      write('Introdu componenta ', j,' : ');
      readln(temp^.s);
```

```
      temp^.next:=p;
      temp^.anti:=NIL;
      p^.anti:=temp;
      p:=temp;
   end;
end;
procedure Afisare_lista(var p:legatura);
var Curent:legatura;
begin
   Curent:=p;
   while Curent<>nil do begin
      writeln(curent^.s);
      curent:=curent^.next;
   end;
end;
BEGIN
   ClrScr;
   write('Introdu n: ');
   readln(n);
   Creeaza_lista(primul);
   writeln('----------------');
   Afisare_lista(primul);
   readkey;
END.
```

❹ Un elev a împrumutat o carte unui alt elev, acesta din urmă (fără consimțământul primului) – altuia ş.a.m.d. Să se afişeze ordinea în care fiecare elev trebuie să returneze cartea, astfel încît ea să ajungă la proprietar, dacă se ştie că fiecare este obligat să înapoieze cartea celui de la care a împrumutat-o.

Rezolvare:

```
program Stiva1; {primul intrat este ultimul iesit}
uses Crt;
type nume=String[20];
     legatura=^persoana;
     persoana=record
           name:nume;
           Next:legatura
     end;
var primul:legatura;
    s:nume;
procedure Creeaza_stiva(var primul:legatura; var s:nume);
begin
   New(primul);
   primul^.Next:=NIL;
   primul^.name:=s
end;
procedure Insereaza_comp(var primul:legatura; var s:nume);
var temp:legatura;
begin
   New(temp);
```

```
   temp^.Next:=primul;
   primul:=temp;
   primul^.name:=s
end;
procedure Scoate_virf(var primul:legatura; var s:nume);
{s este virful scos din stiva}
var temp:legatura;
begin
   temp:=primul;
   s:=primul^.name;
   primul:=primul^.Next;
   dispose(temp);
end;
BEGIN
   Clrscr;
   write('Introdu numele proprietarului: ');
   readln(s);
   Creeaza_stiva(primul,s);
   repeat
      write('Introdu numele urmatorului elev sau scrie STOP: ');
      readln(s);
      Insereaza_comp(primul,s)
   until s='STOP';
   {---- Afisarea continutului stivei ----}
   writeln('Cartea va fi returnata in ordinea: ');
   Scoate_virf(primul,s);{Stergem STOP din stiva}
   repeat
      Scoate_virf(primul,s);
      write(s,'->');
   until primul=NIL;
   write(#8,#8,#8,#8,' ');
   readkey;
END.
```

❺ Se dă numărul natural *n*. Se citesc de la tastatură *n* numere întregi şi se memorizează într-o stivă. Să se creeze altă stivă cu numerele de o cifră ale primei stive.

Rezolvare:

```
program Stiva2;
uses Crt;
type stiva=^celula;
     celula=record
     cimp:integer;
     next:stiva;
end;
var vs1,vs2,p,q:stiva;
    n,i:integer;
BEGIN
   ClrScr;
   write('Introdu n: ');
   readln(n);
```

```
    writeln('=========================');
    for i:=1 to n do begin
       new(p);
       readln(p^.cimp);
       p^.next:=vs1;
       vs1:=p;
    end;
    p:=vs1;
    vs2:=Nil;
    while p<>Nil do begin
       if p^.cimp div 10=0 then begin
          new(q);
          q^.cimp:=p^.cimp;
          q^.next:=vs2;
          vs2:=q;
       end;
       p:=p^.next;
       dispose(vs1);
       vs1:=p;
    end;
    q:=vs2;
    writeln('-------------');
    while q<>Nil do begin
       writeln(q^.cimp);
       q:=q^.next;
    end;
    readkey;
END.
```

❻ *Numărătoarea*

Se dau numerele naturale n şi m, unde $m > 1$. Considerăm m copii care au format un cerc şi unul dintre ei numără într-o direcţie pînă la al m-lea copil care iese din cerc.

Numărătoarea continuă cu următorul jucător.

Ultimul copil rămas mijeşte. Citind de la tastatură numele copiilor şi numărînd de la ultimul copil spre primul, să se determine:

a) ordinea de ieşire din cerc;

b) cine va miji.

De exemplu, pentru grupul:

Ion
Vasile
Petru
Mihai
Ana

şi $m = 3$ se va afişa:

Petru – afara
Ana – afara
Ion – afara
Mihai – afara
Mijeste Vasile

Rezolvare:

```
program lista_ci; {Lista circulara simplu inlantuita}
uses Crt;
type legatura=^Persoana;
     Persoana=record
        Name:string[20];
        Next:legatura
     end;
var  Primul,p,ult,temp:legatura;
     nume:string[20];
     i,n,m:integer;
     c:char;
BEGIN
   ClrScr;
   write('Introdu numarul de copii: ');
   readln(n);
   write('Introdu numarul m: ');
   readln(m);
   write('Scrie numele primului copil: ');
   new(Primul);
   readln(primul^.name);
   ult:=primul;
   for i:=2 to n do begin
      new(p);
      write('Introdu numele copilului ',i,' : ');
      readln(p^.name);
      p^.next:=primul;
      primul:=p;
   end;
   ult^.next:=primul;
   writeln('---- Afisarea listei -----');
   p:=primul;
   repeat
      writeln(p^.name);
      p:=p^.next;
   until p=primul;
   writeln('-----------');
   i:=2; {i este numarul de ordine al urmatorului}
   p:=primul;
   repeat
      if i mod m=0 then begin
         temp:=p^.next;
         p^.next:=p^.next^.next;
         writeln(temp^.name,'- afara');
         dispose(temp);
      end else p:=p^.next;
      i:=i+1;
   until p^.next=p;
   writeln('=========================');
   writeln(' Mijeste ', p^.name);
   readkey;
END.
```

❼ Se dă numărul natural n.

Să se afişeze în ordine crescătoare primele n numere naturale, a căror descompunere în factori primi conţine doar factori din mulţimea $\{2, 3, 5\}$.

Rezolvare:

Construim 3 cozi: $C2$, $C3$ şi $C5$, care vor conţine numere neafişate cu proprietatea menţionată şi care se vor completa după regula de mai jos. Evident, dacă un număr t satisface proprietatea menţionată, atunci numerele $2t$, $3t$, $5t$ de asemenea satisfac această proprietate.

Procedăm astfel:

1. Fie că ultimul număr afişat cu proprietatea menţionată este t (evident primul număr este 1).

2. În coada $C2$ (respectiv $C3$, $C5$) plasăm numărul $2t$ (respectiv $3t$, $5t$).

3. Pentru a afişa următorul număr, alegem dintre vîrfurile cozilor numărul cel mai mic (evident el satisface proprietatea menţionată). Acest număr (fie x) va fi afişat şi va fi scos din coada (eventual cozile) în care a fost găsit. Considerînd $t = x$, se trece din nou la pasul 1.

```
program Coada;
uses Crt;
type legatura=^Comp;
     Comp=record
        numar:longint;
        Next:legatura
     end;
var p2Begin,p2End:legatura; {virful si sfirsitul cozii C2}
    p3Begin,p3End:legatura; {virful si sfirsitul cozii C3}
    p5Begin,p5End:legatura; {virful si sfirsitul cozii C5}
    n1,n2,n3,x,k:longint;
    n,contor:integer;
procedure Creeaza_coada(var pBegin,pEnd:legatura; n:longint);
begin
   New(pBegin);
   pBegin^.Next:=NIL;
   pBegin^.numar:=n;
   pEnd:=pBegin
end;
procedure Adauga_coada(var pEnd:legatura; n:longint);
var temp:legatura;
begin
   New(temp);
   temp^.Next:=NIL;
   pEnd^.Next:=temp;
   pEnd:=temp;
   pEnd^.numar:=n
end;
procedure Citeste_virful(var pBegin:legatura; var n:longint);
begin
   n:=pBegin^.numar;
end;
```

```
procedure Scoate_virful(var pBegin:legatura);
var temp:legatura;
begin
   temp:=pBegin;
   pBegin:=pBegin^.Next;
   dispose(temp);
end;
function min(a,b,c:longint):longint;
var m:longint;
begin
   m:=a;
   if b<m then m:=b;
   if c<m then m:=c;
   min:=m;
end;
procedure scrie_adauga(t:longint);
begin
   inc(contor);
   if contor mod 10=0 then writeln(t) else write(t,' ');
   if t>1 then begin {deoarece pentru t=1, C2, C3, C5 deja au 2, 3, 5}
      Adauga_coada(p2End,2*t);
      Adauga_coada(p3End,3*t);
      Adauga_coada(p5End,5*t);
   end;
end;
BEGIN
   ClrScr;
   write('Introdu n: ');
   readln(n);
   Creeaza_coada(p2Begin,p2End,2);
   Creeaza_coada(p3Begin,p3End,3);
   Creeaza_coada(p5Begin,p5End,5);
   scrie_adauga(1);
   k:=1;
   while k<>n do begin
      Citeste_virful(p2Begin,n1);
      Citeste_virful(p3Begin,n2);
      Citeste_virful(p5Begin,n3);
      x:=min(n1,n2,n3);
      scrie_adauga(x);
      k:=k+1;
      if n1=x then Scoate_virful(p2begin);
      if n2=x then Scoate_virful(p3begin);
      if n3=x then Scoate_virful(p5begin);
   end;
   readkey;
END.
```

❽ Sa se scrie un program care va reduce monoamele asemenea ale unui polinom dc mai multe nedeterminate. Polinomul poate fi oricît de mare, de aceea el se va citi dintr-un fişier *text* cu urmatoarea structură:
- fiecare rînd al fişierului va reprezenta un monom;
- începînd cu prima coloană se va scrie coeficientul monomului, iar partea literală se va scrie începînd cu coloana a opta;
- fiecare nedeterminată a părţii literale se scrie de atîtea ori cît indică exponentul puterii acestei nedeterminate.

De exemplu, pentru polinomul $5ax^3y^2 - 29ax^2y^2 + 11xy + 81$ vom avea următorul conţinut de fişier:

5	*axxxyy*
–29	*axxyy*
11	*xy*
81	

Rezolvare:
Rezultatul execuţiei programului va fi o listă unidirecţională, care mai apoi se va stoca într-un fişier *text* cu aceeaşi structură ca şi a fişierului-sursă. Lista va fi creată pe parcursul reducerii monoamelor asemenea. Citind un monom din fişierul-sursă, se va parcurge lista creată la acel monent şi se va căuta monomul asemenea din listă. Dacă se va găsi un astfel de monom, coeficienţii lor vor fi adunati, altfel – se va mai adăuga o celulă la listă. O celulă a listei va fi ştearsă dacă coeficientul ei este 0.

```
program Polinom;
uses Crt;
type pmonom=^monom;
     monom=record
        coef:integer;
        literala:string; {partea literala}
        next:pmonom;
     end;
var First,B,Curent:pmonom;
    file1,file2:text;
    num,st1,st3:string;
    coef1,i:integer;
function Precedent(P:pmonom):pmonom;
var K:pmonom;
begin
   Precedent:=nil;
   if (P=nil) or (P=First) then Exit;
   K:=First;
   while (K^.Next<>P) do
      if (K^.Next<>nil) then K:=K^.Next
      else Exit;
   Precedent:=K;
end;
function MonomNou(Tliterala:string; Tcoef:integer):pmonom;
var S:Word;
    P:pmonom;
begin
```

```
   S:=SizeOf(Pointer)+SizeOf(Integer)+Length(Tliterala)+1;
   GetMem(P,S);
   with P^ do begin
      literala:=Tliterala;
      coef:=Tcoef;
      next:=nil;
   end;
   MonomNou:=P;
end;
procedure DisposeMonom(P:pmonom);
var S:Word;
    K:pmonom;
begin
   K:=Precedent(P);
   S:=7+Length(P^.literala); {4+2+1}
   if K=nil then First:=P^.Next
      else K^.Next:=P^.Next;
   FreeMem(P,S);
end;
function Cauta(A:string):pmonom;
var P:pmonom;
begin
   P:=First;
   repeat
      if (P^.literala=A) then Break;
      P:=P^.Next;
   until P=nil;
   Cauta:=P;
end;
procedure ReadStr(var F:Text; var S:string; var I:integer);
var St:string;
    B:byte;
    C:integer;
begin
   Readln(F,St);
   for B:=1 to 7 do
      if (St[B+1]=' ') then Break;
   Val(Copy(St,1,B),I,C);
   S:=Copy(St,8,Length(St)-7);
end;
procedure Adauga(A:pmonom);
var P:pmonom;
    S:string;
begin
   S:=A^.literala;
   if (S>First^.literala) then begin
      A^.next:=First;
      First:=A;
      Exit;
   end;
   P:=First;
   repeat
```

```
      if (S>P^.next^.literala) then begin
         A^.next:=P^.next;
         P^.next:=A;
         Exit;
      end;
      P:=P^.next;
   until P=nil;
end;
BEGIN
   ClrScr;
   First:=MonomNou('',0);
   if ParamCount<2 then begin
      writeln('usage: POLINOM <Fisier-sursa> <Fisier-Destinatie>');
      Exit;
   end;
   Assign(file1,ParamStr(1)); {assigneaza file1 cu primul parametru}
   reset(file1);
   while not eof(file1) do begin
      ReadStr(File1,St1,Coef1);
      if Coef1=0 then Continue;
      B:=Cauta(St1);
      if B<>nil then begin
         Inc(B^.Coef, Coef1);
         if (B^.Coef=0) and (B^.literala<>'') then DisposeMonom(B);
      end else Adauga(MonomNou(St1,Coef1));
   end;
   close(file1);
   {pastram lista in fisierul asignat de file2}
   Assign(file2,ParamStr(2));
   rewrite(file2);
   Curent:=First;
   while Curent<>nil do begin
      if Curent^.coef<>0 then begin
         str(Curent^.coef,st3);
         for i:=1 to 7-length(st3) do
            st3:=st3+' ';
         st3:=st3+Curent^.literala;
         writeln(file2,st3);
      end;
      Curent:=Curent^.next;
   end;
   close(file2);
   writeln('Ok!!!');
   readkey;
END.
```

Observație

La apelul programului, în linia de comandă, în afară de numele fișierului executabil, se indică doi parametri: primul specifică numele fișierului-sursă, al doilea – numele fișierului-destinație. Dacă programul se lansează din editorul Turbo Pascal, atunci parametrii se indică în opțiunea *Parameters* a meniului *Run*.

Probleme propuse

A

1. Fie declarațiile:

```
var x,y:^integer;
```

Fie că variabilele *x* și *y* au valorile din desen

Completați:

a) Valoarea variabilei x este _____.

b) Variabila ____ are ca valoare adresa zonei cu întregul ____.

c) _____ sînt variabile dinamice.

d) _____ sînt variabile-referință.

e) Întregul 7 este valoarea _____.

f) După execcutarea secvenței de instrucțiuni `y:=x;` și `x:=y;` va fi următoarea situație:_____

_____.

2. Fie declarațiile:

```
var x,y:^integer;
    z:^real;
    a:integer;
    b:real;
```

Care dintre următoarele instrucțiuni sînt greșite:

a) `x:=a;` b) `b:=x^;` c) `z^:=a;` d) `y^:=addr(a);`

e) `x:=y;` f) `x^:=z^;` g) `z^:=y^;` h) `z:=@b;`

i) `x:=@y;` j) `y:=nil;` k) `x:=@y;` l) `write(p^);`

m) `write(@p)`?

3. Care va fi rezultatul execuției programului, dacă de la tastatură se citește –5 8?

a)
```
program Pointer1;
var x,y,z:^integer;
BEGIN
   new(x);
   new(y);
   readln(x^, y^);
   new(z);
   z^:=x^;
   x^:=y^;
   y^:=z^;
   write(x^, y^);
END.
```

b)
```
program Pointer2;
var x,y:integer;
   z,t:^integer;
```

```
BEGIN
   readln(x, y);
   z:=@x;
   t:=addr(y);
   x:=t^;
   y:=z^;
END.
```

4. Ce va afişa programul?

```
program Pointer3;
var p1,p2:^integer;
    i,k:integer;
BEGIN
   i:=2;
   p1:=@i;
   i:=i+3;
   p2:=@i;
   p^1:=P^1+p^2+i;
   writeln(i);
END.
```

5. Să se depisteze erorile:

a)
```
type m=array[1..10] of integer;
var  p1:^m;
     p2:^string[20];
```

b)
```
type  celula=record
       field1:integer;
       field2:real;
       field3:^celula;
end;
```

c)
```
type  lista=^celula;
      celula=record
        field1:real;
        field2:^set of char;
        field3:^lista;
      end;
```

d)
```
var  lista:^celula;
     celula:record
        filed1:intger;
        field2:^pointer;
        field3;^lista;
     end;
```

6. Să se creeze o listă din 100 de numere aleatoare întregi.

a) Să se determine prin parcurgerea listei suma numerelor pozitive.

b) Să se afişeze numerele în ordinea inversă generării.

c) Să se afişeze numerele în ordinea generării.

7. Să se creeze o listă ce va conține numele, vîrsta şi telefonul a 10 persoane.
   a) Să se afişeze la ecran primele 5 persoane din listă.
   b) Să se afişeze la ecran persoanele de pe poziţiile pare din listă.
   c) Să se afişeze la ecran persoanele cu vîrsta mai mare de 20 de ani.
   d) Să se determine poziţia în listă a persoanei cu numele dat.

8. Pentru fiecare student dintr-o grupă se ştie numele şi 3 note la o sesiune.
   a) Să se creeze lista studenţilor în ordinea lexicografică a numelui.
   b) Să se creeze lista studenţilor în ordinea descrescătoare a notei medii.
   c) Să se creeze lista studenţilor cu nota medie mai mare decît 5 în ordinea lexicografică a numelui.
   d) Să se elimine restanţierii din lista creată în b).

B

9. Să se formuleze problema care se rezolvă cu ajutorul algoritmului:

```
program St;
uses Crt;
type stiva=^celula;
     celula=record
              cimp:integer;
              next:stiva;
     end;
var vs,p,q:stiva;
    n,i:integer;
BEGIN
   ClrScr;
   vs:=Nil;
   readln(n);
   writeln('=========================');
   for i:=1 to n do begin
      new(p);
      readln(p^.cimp);
      p^.next:=vs;
      vs:=p;
   end;
   p:=vs;
   while (p<>Nil) and (p^.cimp mod 3=0) do begin
      p:=p^.next;
      dispose(vs);
      vs:=p;
   end;
   if p=Nil then q:=p else q:=p^.next;
   while q<>Nil do begin
      if q^.cimp mod 3=0 then begin
         p^.next:=q^.next;
         dispose(q);
         q:=p^.next;
      end else begin
         p:=q;
```

```
        q:=q^.next;
      end;
    end;
    p:=vs;
    writeln('----------');
    while p<>Nil do begin
      writeln(p^.cimp);
      p:=p^.next;
    end;
    readkey;
END.
```

10. Utilizînd o stivă, să se afişeze descompunerea zecimală a unui număr întreg dat *n*. De exemplu, pentru $n = -345$ se va afişa $n = -3 * 10 \wedge 2 + 4 * 10 + 5$.

11. Se citesc de la tastatură două şiruri de numere întregi. Sfîrşitul fiecărui şir este marcat prin prezenţa pe poziţia *i* a numărului *i*.
a) Să se formeze al treilea şir format din elementele primelor două şiruri.
b) Să se ordoneze crescător elementele şirului al treilea.

12. a) Se citesc de la tastatură două şiruri de numere întregi mai mici decît 1 000 pînă cînd se citeşte un alt fel de număr şi se formează două liste-mulţimi. Să se afişeze mulţimile.
b) Să se afişeze reuniunea şi intersecţia mulţimilor din a).

13. Se dă un fişier *text*.
a) Să se formeze lista cuvintelor din text în ordinea alfabetică.
b) Să se calculeze numărul cuvintelor de lungime maximală.

14. Se consideră o listă de cuvinte. Să se scrie un subprogram:
a) pentru inserarea unui cuvînt la începutul listei;
b) pentru inserarea unui cuvînt la sfîrşitul listei;
c) pentru verificarea existenţei unui cuvînt în listă;
d) pentru ştergerea unui cuvînt din listă.

15. Se dă un fişier text. Să se afişeze la ecran „răsturnatul" conţinutului fişierului.

16. Se dă o listă de numere întregi. Să se scrie un subprogram care din lista dată formează două liste: una formată din numerele negative, alta – din celelelate numere.

17. Să se genereze două liste de numere întregi.
a) Să se creeze a treia listă formată din numerele care se întîlnesc doar în una dintre primele două liste.
b) Să se creeze a treia listă concatenînd primele două liste şi ordonînd crescător componentele ei.

18. Se dau două liste de numere întregi.
a) Să se scrie un subprogram care verifică dacă listele sînt egale.
b) Să se scrie un subprogram care verifică dacă o listă se conţine în alta.
c) Să se scrie o funcţie care returnează numărul de componente comune ale listelor.

19. Funcţia `OFS (v)` returnează numărul de octeţi ce reprezintă adresa variabilei `v`, relativ de segmentul în care se află ea. Se dă o matrice de numere întregi. Să se arate că elementele matricei $A(m, n)$ se scriu în memorie în ordinea
$a_{11}, a_{12}, ..., a_{1n}, a_{22}, ..., a_{2n}, ..., a_{mn}$.

20. Un polinom de o singură nedeterminată se poate reprezenta printr-o listă, unde fiecare celulă a listei este un monom al polinomului. Să se realizeze un algoritm ce efectuează:
a) adunarea a două polinoame;
b) scăderea a două polinoame;
c) înmulţirea a două polinoame.

21. Un număr natural mare poate fi reprezentat printr-o listă, unde fiecare celulă a listei conţine o cifră a numărului. Să se realizeze un algoritm care va efectua:
a) adunarea a două numere naturale mari date;
b) scăderea a două numere naturale mari date;
c) înmulţirea a două numere naturale mari date.

22. Să se realizeze un algoritm care va efectua înmulţirea a două polinoame de mai multe nedeterminate. Polinoamele se vor citi din fişiere *text*.
Indicaţie. Observaţi rezolvarea problemei 7 din secvenţa *Probleme rezolvate*.

14 Unit-uri proprii

Sugestii teoretice

Structura generală a unui unit

Antetul unit-ului	`Unit nume_unit`	

Secţiunea de interfaţă	`Interface`
	`  uses ...`
	`  const ...`
	`  type ...`
	`  var ...`
	`  antete de subprograme`
		Obiecte globale

Secţiunea de implementare	`Implementation`
	`  uses ...`
	`  label ...`
	`  const ...`
	`  type ...`
	`  subprograme locale`
	`  subprograme definite în Interface`
		Obiecte locale

Secţiunea de iniţializare (poate să lipsească)	`Begin`	
	`End.`	

Observaţii

1. Unit-ul reprezintă o colecţie de resurse de program, destinate spre utilizare de către alte unit-uri sau programe.

2. Textul-sursă din care se va obţine unit-ul se salvează sub acelaşi nume ca şi cel precizat în antetul unit-ului. În urma compilării unit-ului se va obţine un nou fişier cu extensia .tpu cu numele precizat în antet. Înainte de compilare trebuie să fie activată opţiunea *Destination Disk* din submeniul *Compile* al meniului principal al editorului Turbo Pascal.

3. În secţiunea *Interface* sînt descrise obiectele globale, adică obiectele vizibile (accesibile) din alte unit-uri sau programe. În secţiunea *Implementation* sînt plasate obiectele de ordin local (obiectele ascunse), utilizate pentru organizarea resurselor unit-ului.

Observații

4. În *Interface* se scrie doar antetul subprogramelor de ordin global, descrierea lor fiind plasată în secțiunea *Implementation*. De regulă, în *Implementation* antetul subprogramului de ordin global se scrie fără lista parametrilor formali. Se poate scrie și antetul complet, ținînd seama ca el să coincidă cu cel din *Interface*.

5. Pentru a utiliza resursele unui unit U_1 în alt unit U_2 (sau program P), numele unitului U_1 se scrie după directiva *Uses* a unitului U_2 (respectiv a programului P).

Problemă rezolvată

a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor aritmetice (adunarea, înmulțirea, împărțirea) cu numerele complexe.

b) Utilizînd resursele unit-ului creat în a), să se calculeze suma, produsul, cîtul numerelor complexe 3 – 4i și 2 + 3,5i.

Rezolvare:

a) Amintim că dacă $x = a + bi$ și $y = c + di$, atunci:

$x + y = (a + c) + (b + d)$i;

$x \cdot y = (ac - bd) + (ad + bc)$i;

$\frac{x}{y} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}$i.

```
Unit Compl;
INTERFACE
   type Complex=record {tipul complex va fi global}
        Re,Im:real;
   end;
   procedure InitC(var C:Complex; R,I:real);
   {initializeaza numarul complex C}
   procedure AdunC(C1,C2:Complex; var R:Complex);
   procedure InmultC(C1,C2:Complex; var R:Complex);
   procedure ImpartC(C1,C2:Complex; var R:Complex);
   procedure ScrieC(C:Complex);
   {Afiseaza la ecran numarul complex C in forma a+bi}
IMPLEMENTATION
   procedure InitC;
   begin
      C.Re:=R; C.Im:=I;
   end;
   procedure AdunC;
   begin
      R.Re:=C1.Re+C2.Re;
      R.Im:=C1.Im+C2.Im;
   end;
   procedure InmultC;
   begin
      R.Re:=C1.Re*C2.Re-C1.Im*C2.Im;
      R.Im:=C1.Re*C2.Im+C1.Im*C2.Re;
   end;
```

```
  procedure ImpartC;
  var t:real;
  begin
     t:=sqr(C2.Re)+sqr(C2.Im);
     R.Re:=(C1.Re*C2.Re+C1.Im*C2.Im)/t;
     R.Im:=(C1.Im*C2.Re-C1.Re*C2.Im)/t;
  end;
  procedure ScrieC;
  begin
     with C do begin
        write(Re:2:2);
        if Im=0 then exit;
        if Im>0 then write('+');
        write(Im:2:2,'i');
     end;
  end;
END.
```

b) Înainte de a scrie algoritmul pentru realizarea adunării, înmulțirii și împărțirii celor două numere complexe date, vom:

- salva fișierul unit-ului sub numele Compl.pas;
- compila unit-ul Compl, activînd în prealabil *Compile/Destination Disk*. În urma acestei operații pe disc se va crea fișierul Compl.tpu.

```
program Apl_unit;
uses Crt, Compl;
var x,y,z:Complex;
BEGIN
   ClrScr;
   InitC(x,3,-4);
   InitC(y,2,3.5);
   AdunC(x,y,z); {z=x+y}
   ScrieC(x); write(' + '); ScrieC(y); write(' = ');
   WriteC(Z); writeln;
   InmultC(x,y,z); {z=x*y}
   ScrieC(x); write(' * '); ScrieC(y); write(' = ');
   WriteC(z); writeln;
   ImpartC(x,y,z); {z=x/y}
   ScrieC(x); write(' : '); ScrieC(y); write(' = ');
   readkey;
END.
```

Probleme propuse

A

1. Completați unit-ul Compl creat în exemplul rezolvat cu subprograme pentru:
   - determinarea conjugatului unui număr complex;
   - calcularea modulului numărului complex;
   - calcularea argumentului principal al numărului complex (mărimea unghiului format de

vectorul $\overrightarrow{OM}$ şi semiaxa pozitivă Ox, unde $\overrightarrow{OM}$ este reprezentarea geometrică a numărului complex).

- calcularea unei rădăcini de ordinul n a unui număr complex.

2. a) Să se creeze un unit care va conţine subprograme pentru calcularea valorilor funcţiilor:

- tangenta;
- cotangenta;
- arscsinus;
- arccosinus;
- arccotangenta;
- hiperbolice;
- logaritmului cu baza diferiră de numărul e.

Indicaţii. Pentru definirea funcţiilor trigonometrice inverse să se consulte tema 1.

Funcţii hiperbolice:

$\text{Sh } x = \dfrac{e^x - e^{-x}}{2}$ – sinusul hiperbolic;

$\text{Ch } x = \dfrac{e^x + e^{-x}}{2}$ – cosinusul hiperbolic;

$\text{Th } x = \dfrac{e^x + e^{-x}}{e^x + e^{-x}}$ – tangenta hiperbolică;

$\text{Cth } x = \dfrac{e^x + e^{-x}}{e^x - e^{-x}}$ – cotangenta hiperbolică.

b) Să se calculeze valoarea fiecărei funcţii definite în a) pentru argumentul dat x.

3. a) Să se creeze un unit care va conţine subprograme pentru calcularea:

- lungimii unui cerc, ariei discului marginit de acest cerc fiind dată raza cercului;
- razei cercului, fiind dată lungimea lui sau aria discului mărginit de acest cerc.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

4. a) Să se creeze un unit care va conţine subprograme pentru:

- transformarea măsurii unui unghi din grade în radiani, şi invers;
- calcularea sumei măsurilor a două unghiuri;
- calcularea diferenţei măsurilor a două unghiuri;
- compararea măsurilor a două unghiuri.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

5. a) Să se creeze un unit care va conţine subprograme pentru prelucrarea tablourilor unidimensionale (vectorilor) de numere întregi:

- citirea tabloului;
- afişarea tabloului;
- aflarea componentei maximale a tabloului;
- aflarea componentei minimale a tabloului;
- ordonarea crescătoare a componentelor tabloului;

- ordonarea descrescătoare a componentelor tabloului;
- calcularea mediei aritmetice a componentelor tabloului.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

6. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor cu matrice:
- citirea matricei;
- afișarea la ecran a matricei;
- adunarea a două matrice;
- înmulțirea a două matrice.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

7. a) Să se creeze un unit care va conține subprograme pentru rezolvarea triunghiului, fiind date trei elemente (laturi, unghiuri) ale acestuia, dintre care cel puțin o latură. Prin expresia *a rezolva un triunghi* vom înțelege aflarea măsurilor unghiurilor, lungimilor laturilor, medianelor, bisectoarelor, înălțimilor triunghiului.

b) Să se realizeze un program care va rezolva un triunghi, fiind date trei elemente (laturi, unghiuri) ale acestuia, dintre care cel puțin o latură.

8. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor (adunarea, scăderea, înmulțirea, împărțirea, simplificarea, ridicarea la putere naturală, compararea) cu fracții.

b) Utilizînd subprogramele unit-ului creat în a), să se afișeze fracția ireductibilă – valoarea fiecărei expresii:

$$E_1 = \frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\frac{1}{8}+\frac{1}{9}+\frac{1}{10};$$

$$E_2 = 1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\frac{1}{6}+\frac{1}{7}-\frac{1}{8}+\frac{1}{9}-\frac{1}{10};$$

$$E_3 = \frac{1}{2}*\frac{1}{3}*\frac{1}{4}*\frac{1}{5};$$

$$E_4 = \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{3}}}}}.$$

c) Se dă un vector cu n, $1 < n < 100$, componente-fracții. Afișați componentele vectorului în ordine crescătoare.

9. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor cu vectori în plan:
- adunarea vectorilor;
- înmulțirea vectorilor cu un scalar;
- calcularea produsului scalar a doi vectori;

- determinarea lungimii vectorului;
- verificarea coliniarității a doi vectori.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

10. a) Să se creeze un unit care va conține subprograme pentru prelucrarea numerelor naturale:
- determinarea parității numărului;
- determinarea succesorului numărului;
- determinarea predecesorului numărului;
- determinarea numărului de cifre ale numărului;
- determinarea cifrelor numărului;
- determinarea cifrei de pe poziția indicată;
- verificarea existenței unei cifre date în scrierea numărului;
- determinarea divizorilor numărului;
- determinarea apartenenței numărului la mulțimea numerelor prime;
- scrierea numărului ca produs de factori primi.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

11. a) Să se creeze un unit care va conține subprograme pentru prelucrarea perechilor de numere naturale:
- stabilirea numărului mai mare;
- calcularea sumei (diferenței, produsului) numerelor;
- determinarea celui mai mare divizor comun al numerelor;
- determinarea celui mai mic multiplu comun al numerelor;
- determinarea cifrelor comune ale numerelor.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

B

12. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor arirmetice (adunarea, scăderea, înmulțirea, împărțirea) cu numere lungi (în care numărul de cifre poate fi 100).

b) În baza unitului creat în a), să se elaboreze programul „Calculator", care va efectua operații aritmetice cu numere lungi.

13. a) Să se creeze un unit care va conține subprograme pentru:
- determinarea numărului de cuvinte dintr-un string;
- substituirea unui cuvînt prin alt cuvînt într-un string;
- determinarea numărului de apariții ale unui cuvînt într-un string.

b) Se dă un fișier *text*. Utilizînd subprogramele unit-ului creat în a):
- să se determine rîndul din fișier cu cele mai multe cuvinte;
- să se substituie în tot fișierul un cuvînt prin alt cuvînt;
- să se determine numărul de apariții ale unui cuvînt în fișier;
- să se determine cuvîntul care apare de cele mai multe ori în fișier.

14. a) Se consideră un sistem de axe ortogonale din plan. Să se creeze un unit care va conține subprograme pentru:
- definirea unui punct;
- definirea unei drepte;
- definirea unui cerc;
- determinarea poziției relative a unui punct față de o dreaptă;
- determinarea poziției relative a unui punct față de un cerc;
- determinarea poziției relative a unei drepte față de un cerc;
- determinarea poziției relative a două drepte;
- determinarea poziției relative a două cercuri.

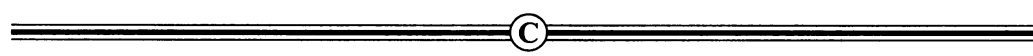
b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

15. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor cu mulțimi de numere naturale mai mici decît 1 000:
- definirea unei mulțimi;
- afișarea la ecran a unei mulțimi;
- determinarea cardinalului unei mulțimi;
- reuniunea a două mulțimi;
- intersecția a două mulțimi;
- verificarea incluziunii a două mulțimi;
- determinarea complementarei unei mulțimi în raport cu altă mulțime.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

16. a) Să se creeze un unit care va conține subprograme pentru realizarea operațiilor cu polinoame de o nedeterminată:
- definirea unui polinom;
- determinarea gradului polinomului;
- verificarea egalității a două polinoame;
- adunarea a două polinoame;
- înmulțirea a două polinoame;
- CMMDC a două polinoame;
- împărțirea (cu rest) a două polinoame;
- calcularea valorii polinomului;
- ridicarea la putere a unui polinom;
- determinarea derivatei polinomului.

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).



17. a) Să se creeze un unit care va conține subprograme pentru prelucrarea structurilor de date *listă simplu înlănțuită* (crearea listei, inserarea unei celule în listă, ștergerea unei celule, afișarea celulei de pe poziția indicată).

b) Să se realizeze un program care va utiliza subprogramele unit-ului creat în a).

15 Posibilităţi grafice

Sugestii teoretice

Iniţializarea regimului grafic

În momentul lansării în execuţie a unui program din Turbo Pascal ecranul se află în regim textual, de aceea orice algoritm ce utilizează mijloace grafice trebuie să conţină o secvenţă de iniţializare a regimului grafic. Pentru iniţializare se foloseşte procedura **InitGraph**(**var** *gd*, *gm*: integer; *gpath*: string), unde *gd* determină tipul driver-ului grafic (fişier cu extinderea .bgi), *gm* stabileşte regimul de lucru al adapteru-lui grafic, *gpath* conţine numele driver-ului şi calea pînă la el.

Dacă driver-ul se află în catalogul curent, atunci *gpath* poate fi un şir vid.

Observaţie

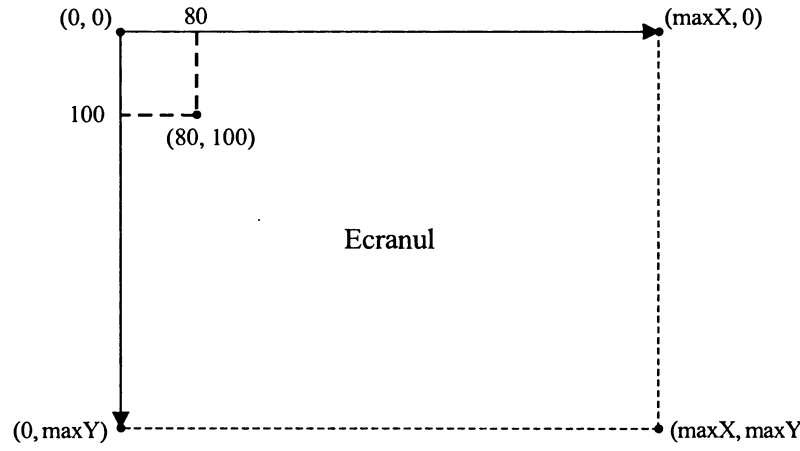
Deseori variabilei *gd* i se atribuie valoarea 0, corespunzătoare regimului de autodeterminare a tipului.

Sistemul de coordonate

Observaţii

1. Spre deosebire de regimul textual, în regim grafic indicatorul poziţiei curente este invizibil.

2. Poziţia indicatorului se determină de coordonatele orizontalei şi verticalei relativ de colţul stînga-sus. Coordonatele sînt numere întregi pozitive.



Funcția **GetMaxX**: `word` returnează maxX – coordonata maximală a orizontalei.
Funcția **GetMaxY**: `word` returnează maxY – coordonata maximală a verticalei.
Funcția **GetX**: `integer` returnează coordonata orizontalei indicatorului.
Funcția **GetY**: `integer` returnează coordonata verticalei indicatorului.

Unele subprograme ale unit-ului Graph

Observație

Biblioteca Graph conține peste 50 de subprograme. Aici sînt prezentate doar cîteva (cele care se vor utiliza în exemple). Toate constantele, variabilele, tipurile, procedurile și funcțiile bibliotecii Graph sînt prezentate în Anexa 2 de la sfîrșitul cărții.

Funcția **GraphRezult**: `integer` returnează codul rezultatului ultimei adresări către o procedură grafică. Codul 0 (sau constanta grOk) este returnat în cazul în care această adresare a decurs reușit. Celelalte coduri sînt valori ale mulțimii $\{-14, -13, -12, ..., -1\}$.

Funcția **GraphErrorMsg** `(Cod:Integer):string` returnează mesajul corespunzător valorii `Cod`, returnată de funcția `GraphResult`.

Procedura **CloseGraph** închide regimul grafic și restabilește regimul textual al ecranului. Memoria ocupată de driver-ul grafic se eliberează.

Procedura **RestoreCrtMode** oprește regimul grafic și restabilește regimul textual, neeliberînd memoria ocupată de driver-ul grafic.

Procedura **SetGraphMode** `(mod:integer)` – stabilește un nou regim grafic, al cărui cod este valoarea `mod`.

Funcția **GetGraphMode**: `integer` returnează codul regimului grafic curent.

Procedura **MoveTo** `(x,y:integer)` stabilește noile coordonate ale indicatorului.

Procedura **MoveRel** `(dx,dy:integer)` atribuie creșterea `dx` (respectiv `dy`) coordonatei orizontalei (respectiv verticalei) indicatorului.

Procedura **ClearDevice** curăță ecranul.

Procedura **PutPixel** `(x,y:integer;c:word)` construiește punctul de coordonatele `(x,y)` și de culoarea `c`.

Procedura **Line** `(x1,y1,x2,y2:integer)` construiește un segment cu extremitățile `(x1,y1)` și `(x2,y2)`.

Procedura **LineTo** `(x,y:integer)` construiește un segment cu o extremitate `(x,y)` și alta – poziția indicatorului.

Procedura **LineRel** `(dx,dy:integer)` construiește un segment cu extremitățile `(indx,indy)` și `(indx+dx,indx+dy)`, unde `(indx,indy)` sînt coordonatele indicatorului.

Procedura **Rectangle** `(x1,y1,x2,y2:integer)` construiește un dreptunghi, ale cărui vîrfuri stînga-sus și dreapta-jos vor avea respectiv coordonatele `(x1,y1)` și `(x2,y2)`.

Procedura **Circle** `(x,y:integer;r:word)` construiește un cerc de rază `r` și centru `(x,y)`.

Procedura **Arc**(x,y:integer;ustart,ufinal,r:word) construieşte un arc de cerc de rază r şi centru (x,y), cu extremităţile ustart şi ufinal, exprimate în grade. Unghiul de 0° corespunde direcţiei orizontale (de la stînga spre dreapta). Unghiurile se depun în direcţia opusă mişcării acelor ceasornicului.

Procedura **Elipse**(x,y:integer,ustart,ufinal,rx,ry:word) construieşte un arc de elipsă, cu centrul (x,y), razele rx, ry, extremităţile ustart, ufinal.

Procedura **SetColor**(c:word) stabileşte culoarea c de bază pentru linii (c∈{0, 1, ..., 15}).

Procedura **SetBkColor**(c:word) stabileşte culoarea c pentru fundal.

Funcţia **ImageSize**(x1,y1,x2,y2:integer):word returnează numărul de octeţi necesari pentru păstrarea în memorie a fragmentului dreptunghiular de ecran, ale cărui vîrfuri stînga-sus şi dreapta-jos au coordonatele (x1,y1) şi (x2,y2).

Procedura **GetImage**(x1,y1,x2,y2:integer;var p) păstrează în memorie fragmentul dreptunghiular de ecran, ale cărui vîrfuri stînga-sus şi dreapta -jos au coordonatele (x1,y1) şi (x2,y2). Variabila p este de tip pointer.

Procedura **PutImage**(x,y:integer;var p; mod:word) plasează fragmentul dreptunghiular de ecran, memorizat anterior cu GetImage, în p.

Colţul stînga-sus va avea coordonatele (x,y).

Mod este modul plasării şi poate avea una din valorile 0, 1, ..., 4.

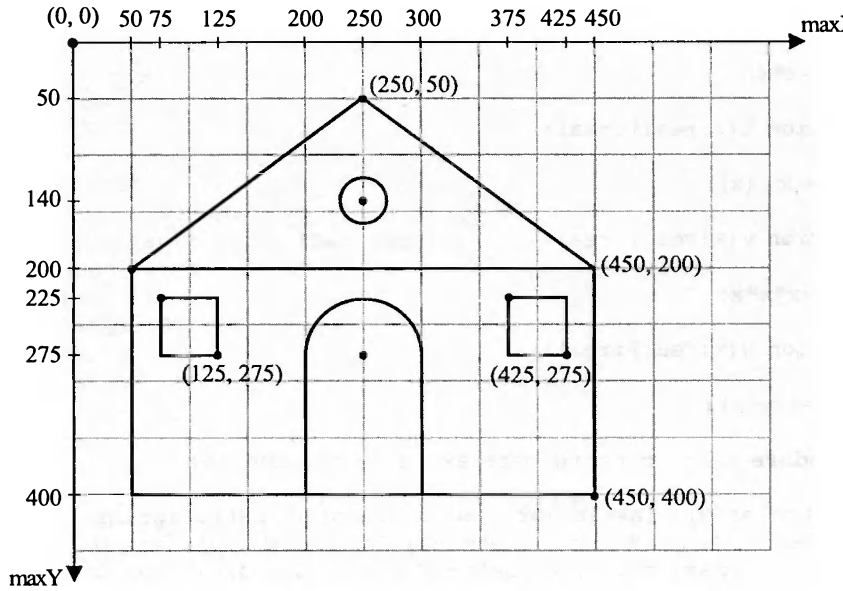
Procedura **OutText**(t:string) afişează orizontal textul t din poziţia indicatorului.

Procedura **OutTextxy**(x,y:integer;t:string) afişează orizontal textul t din poziţia (x,y).

Probleme rezolvate

❶ Următorul program desenează o căsuţă.

```
program Graph1;
uses Crt,Graph;
procedure initiere;
var gd,gm:integer;
    ErrCode:integer;
begin
   gd:=Detect; {autodeterminarea driver-ului}
   InitGraph(gd, gm,''); {initializarea regimului grafic}
   ErrCode := GraphResult; {obtinerea codului rezultatului}
   if ErrCode<>grOk then {verificarea codului}
      writeln('Graphics error:',GraphErrorMsg(ErrCode)); {afisarea
         mesajului corespunzator erorii}
end;
BEGIN
   initiere;{initializarea regimului grafic}
   Rectangle(50,200,450,400);
   moveto(50,200);
```




```
   lineto(250,50);
   lineto(450,200);
   circle(250,140,20);
   rectangle(75,225,125,275);
   rectangle(375,225,425,275);
   line(200,400,200,275);
   line(300,400,300,275);
   arc(250,275,0,180,50);
   readkey;
   CloseGraph;
END.
```

❷ Sa se construiască în acelaşi sistem de axe ortogonale şi pe acelaşi interval $[a, b]$ dat graficul funcţiilor $y = x^2$, $z = \cos x$, $v = x^3$, $w = e^x$.

Rezolvare:

```
program Graph2;
{$F+} {Conectam regimul de recunoastere de la distanta a parametrilor
   de tip functie}
uses Crt,Graph;
type tipF=function(x:real):real;
var a,b:real;{graficul se va construi pe intervalul [a,b]}
procedure initiere;
var gd, gm:integer;
    CodEroare:integer;
begin
   gd:= Detect;
   InitGraph(gd, gm,'');
   CodEroare:=GraphResult;
   if CodEroare<>grOk then writeln(GraphErrorMsg(CodEroare));
end;
```

```
function y(x:real):real;
begin
   y:=x*x;
end;
function z(x:real):real;
begin
   z:=cos(x);
end;
function v(x:real):real;
begin
   v:=x*x*x;
end;
function w(x:real):real;
begin
   w:=exp(x);
end;
procedure axe; {construieste axele de coordonate}
begin
   setcolor(9); {axele vor avea culoarea albastru-aprins}
   line(0, GetmaxY div 2, GetmaxX, GetmaxY div 2);{axa Ox}
   line(GetmaxX div 2, 0, GetmaxX div 2, GetmaxY);{axa Oy}
end;
procedure grafic(f:tipF; m,n:real; c:word);
{construieste graficul functiei f pe intervalul [m, n] cu culoarea c}
var x:real;
begin
   x:=m;
   while x<=n do begin
      putpixel(round(20*x)+GetmaxX div 2,round(-f(x)*20)+GetmaxY div 2,c);
      x:=x+0.0001;
   end;
end;
{$F-}
BEGIN
   ClrScr;
   write('Introdu extremitatile intervalului: ');
   readln(a,b);
   initiere;
   axe;
   grafic(y,a,b,7);{graficul va avea culoarea sur-deschis}
   grafic(z,a,b,10);{graficul va avea culoarea verde-aprins}
   grafic(v,a,b,11);{graficul va avea culoarea cyan-aprins}
   grafic(w,a,b,13);{graficul va avea culoarea zmeurie}
   readkey;
   closegraph;
END.
```

❸ Să se modeleze mişcarea unei bărci pe apă.

Rezolvare:

Observaţie

Barca se va mişca de la stînga spre dreapta. Vom utiliza un ciclu care va depune imaginea. Fiecare punct al imaginii se va mişca pe o sinusoidă, astfel creîndu-se impresia plutirii pe val.

```
program Graph3;
uses Graph,Crt;
var Gd,Gm,i:integer;
   P:pointer;
   Size:word;
BEGIN
   Gd:=Detect;
   InitGraph(Gd, Gm, '');
   if GraphResult<>grOk then Halt(1);
   {Inceput desenare}
   MoveTo(2,2);
   LineRel(30,30);
   LineRel(70,0);
   LineRel(30,-30);
   LineRel(-130,0);
   {Sfirsit desenare}
   Size:=ImageSize(0,0,135,35);
   GetMem(P,Size); {Alocarea memoriei in heap}
   GetImage(0,0,135,35,P^);
   {Fereastra de ecran se pastreaza in P}
   Readln;
   ClearDevice;
   for i:=1 to GetmaxX-130 do begin
      PutImage(i,getmaxY div 2+round(20*sin(0.1*i)),P^,NormalPut);
   Delay(500);
   end;
   CloseGraph;
END.
```

❹ *Rezolvăm şi ne jucăm*

Următorul program desenează un purceluş, care poate fi deplasat pe ecran cu ajutorul tastelor-săgeţi, precum şi cu ajutorul tastelor Home (dreapta-sus), PageDown (dreapta-jos), PageUp (stînga-sus), End (stînga-jos). Ieşirea din execuţia programului are loc în urma apăsării tastei ESC.

```
program Graph4;
uses Crt,Graph;
var size:word;
    p:pointer;
    curentX,CurentY:integer;
procedure initiere;
var gd,gm:integer;
    CodEroare:integer;
begin
   gd:=detect;
   InitGraph(gd,gm,'');
   CodEroare:=GraphResult;
   if CodEroare<>grOk then
      writeln('Graphics error:',GraphErrorMsg(CodEroare));
end;
procedure Purcel; {deseneaza purcelul}
begin
   circle(100,100,20);{capul}
```

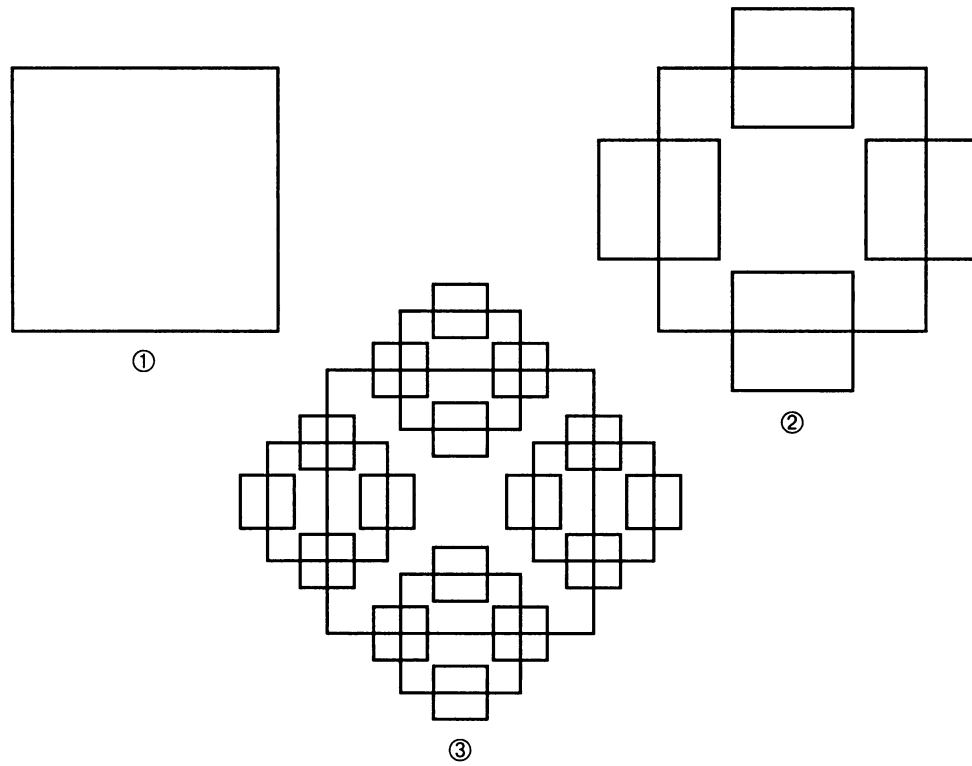
```
   circle(100,105,8);{ritul}
   circle(97,105,1);{nara stinga}
   circle(103,105,1);
   arc(100,113,180,360,2);{gura}
   circle(90,95,2);{ochiul sting}
   circle(110,95,2); {ochiul drept}
   arc(135,115,90,128,40);{urechea dreapta}
   arc(95,75,312,360,40);
   arc(65,115,52,90,40);{urechea stinga}
   arc(105,75,180,228,40);
   ellipse(150,110,0,144,50,42);{corpul}
   ellipse(150,110,195,360,50,40);
   line(118,140,116,160);
   moveTo(116,160);
   LineRel(3,-2);
   LineRel(3,2);
   LineRel(0,-20);
   line(158,140,160,160);
   moveTo(158,160);
   LineRel(3,-2);
   LineRel(3,2);
   LineRel(0,-20);
   arc(210,100,180,360,10); {coada}
end;
procedure Save_purcel; {salveaza imaginea cu purcelul in variabila p^}
begin
   size:=ImageSize(50,50,230,180);
   getMem(p,size);
   GetImage(50,50,230,180,p^);
end;
procedure Misca_purcel(dx,dy:integer);
{deplaseaza imaginea cu Dx pixeli pe orizontala si Dy pixeli pe verticala}
var X,Y:word;
begin
   X:=curentX+dx;
   Y:=curentY+dy;
   if (X>0) and (x<GetMaxX-180) and (Y>0) and (Y<GetMaxY-130) then begin
      putImage(X,Y,P^,NormalPut);
      curentX:=X;
      curentY:=Y;
   end;
end;
procedure Misca; {deplaseaza imaginea pina se va apasa tasta Esc}
var Stop:boolean;
    const D=5;{pasul miscarii}
begin
   Stop:=False;
   repeat
      case Readkey of {# este echivalentul lui chr}
         #27: Stop:=True; {Esc}
         #71: Misca_purcel(-D,-D); {stinga-sus}
         #72: Misca_purcel(0,-D); {sus}
         #73: Misca_purcel(D,-D); {dreapta-sus}
```

```
        #75: Misca_purcel(-D,0); {stinga}
        #77: Misca_purcel(D,0); {dreapta}
        #79: Misca_purcel(-D,D); {stinga-jos}
        #80: Misca_purcel(0,D); {jos}
        #81: Misca_purcel(D,D); {dreapta-jos}
      end
    Until Stop
  end;
  BEGIN
    initiere;
    purcel;
    Save_purcel;
    ClearDevice;
    PutImage(1,1,P^,normalPut);
    curentX:=1; {pozitia initiala}
    curentY:=1;
    misca;
    closegraph;
  END.
```

❺ Figura ③ este construită din pătratul ① în felul următor:

1. Pe fiecare latură a pătratului se construieşte un pătrat, astfel încît centrul lui coincide cu mijlocul laturii pătratului ① şi latura noului pătrat este egală cu 45% din latura pătratului ①. Se obţine figura ②.

2. Pasul **1** se repetă cu fiecare dintre cele 4 pătrate obţinute.



Să se realizeze un algoritm care, utilizînd un subprogram recursiv, va construi figuri ca în desen.

Rezolvare:

```
program Graph5; {Grafica & Recursie}
uses Crt,Graph;
const p=20;
var a,x,y:integer; {x,y sint coordonatele centrului patratului, iar
                    a este jumatate de latura}
procedure initiere;
var gd,gm:integer;
    CodEroare:integer;
begin
   gd:=Detect;
   InitGraph(gd, gm,'');
   CodEroare:=GraphResult;
   if CodEroare<>grOk then
     writeln('Graphics error:',GraphErrorMsg(CodEroare));
end;
procedure patrat(x,y,a:integer); {subprogramul recursiv}
begin
   if a>p then begin
      patrat(x,y-a,round(0.45*a));
      patrat(x+a,y,round(0.45*a));
      patrat(x,y+a,round(0.45*a));
      patrat(x-a,y,round(0.45*a));
   end;
   rectangle(x-a,y-a,x+a,y+a);
end;
BEGIN
   initiere;
   patrat(GetMaxX div 2,GetMaxY div 2,120);
   readkey;
   closegraph;
END.
```

6 Utilizînd generatorul de numere aleatoare, să se modeleze „balonaşe de săpun". Apar aleator puncte care se transformă în cercuri (baloane). Acestea din urmă, continuînd să „crească", se sparg (de asemenea aleator).

Rezolvare:

```
program Baloane;
uses crt,graph;
type balon=record
        x,y,i:integer;
        max:integer;{max este raza maximala a balonului}
     end;
var k,l:integer;
    a:array[1..200] of balon;
procedure initiere;
var gd,gm,errcode:integer;
begin
   gd:=detect;
```

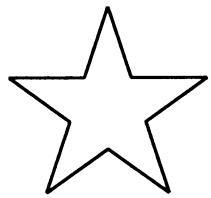
```
   initgraph(gd,gm,'c:\tp\bgi');
   errcode:=graphresult;
   if errcode<>grok then write('eroare');
end;
BEGIN
   initiere;
   randomize;
   for k:=1 to 200 do begin
      a[k].x:=random(getmaxx);
      a[k].y:=random(getmaxy);
      a[k].max:=random(30)+10;
      a[k].i:=random(100)-100;
   end;
   repeat
      cleardevice;
      for k:=1 to 200 do
         if (a[k].i>=0) and (a[k].i<a[k].max) then
            circle(a[k].x,a[k].y,a[k].i)
            else begin
               a[k].x:=random(getmaxx);
               a[k].y:=random(getmaxy);
               a[k].i:=random(100)-100;
               a[k].max:=random(30)+10;
            end;
      delay(2000);
      for k:=1 to 200 do inc(a[k].i);
   until keypressed;
   closegraph;
END.
```

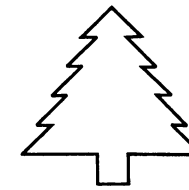
Probleme propuse

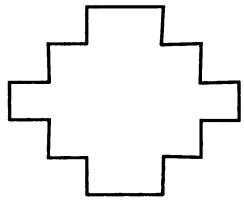
A

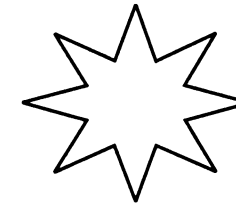
1. Să se elaboreze un algoritm care va desena:
   a) un dreptunghi;
   b) un triunghi dreptunghic;
   c) un cerc;
   d) un semicerc;
   e) un arc de cerc de $60°$;
   f) un paralelogram cu două laturi orizontale;
   g) un paralelogram cu două laturi verticale;
   h) un paralelogram cu laturile oblice;
   i) o coroană circulară;
   j) un sector de cerc;
   k) un triunghi regulat;
   l) un hexagon regulat;
   m) o elipsă.

2. Să se elaboreze un algoritm care va desena următorul desen:

a) 

b) 

c) 

d) 

3. Să se elaboreze un algoritm care va desena:
a) un cub;
b) un paralelipiped;
c) o piramidă;
d) un cilindru;
e) un con circular drept;
f) un trunchi de con circular drept.

4. Să se elaboreze ~~un~~ algoritm care va construi în acelaşi sistem de axe ortogonale şi pe acelaşi interval dat $[a, b]$ graficele funcţiilor $y = x^4$, $g = 2x^2 + 3x - 4$, $h = \sqrt[3]{x}$.

5. Să se elaboreze un algoritm care va construi în regim grafic tabla de şah.

6. Să se elaboreze un algoritm care va desena pe tot ecranul:
a) litera T; b) litera O; c) litera A; d) litera B; e) litera C; f) litera X.

7. Să se elaboreze un algoritm care va desena pe tot ecranul:
a) cifra 0; b) cifra 1; c) cifra 2; d) cifra 3; e) cifra 4;
f) cifra 5; g) cifra 6; h) cifra 7; i) cifra 8; j) cifra 9.

8. Se citesc de la tastatură coordonatele unui punct M. Să se deseneze aleator 10 drepte concurente în punctul M.

9. Să se modeleze mişcarea unei rachete.

10. Să se modeleze mişcarea acelor clasornicului.

11. Utilizînd generatorul numerelor aleatoare, să se modeleze în regim grafic „cerul înstelat".

12. Să se elaboreze un algoritm care va afişa aleator puncte de diferite culori pe toată suprafaţa ecranului în afară de:
a) interiorul dreptuinghiului cu extremităţile unei diagonale în punctele de coordonate (100, 100) şi (400, 300);
b) interiorul cercului de centru (200, 200) şi rază 50;
c) interiorul pătratelor *ABCD* şi *MNKP*, unde $A(0, 0)$, $C(100, 100)$, $M(300, 300)$, $P(400, 400)$.

13. Să se modeleze mişcarea unui punct:
a) pe un cerc; b) pe o elipsă.

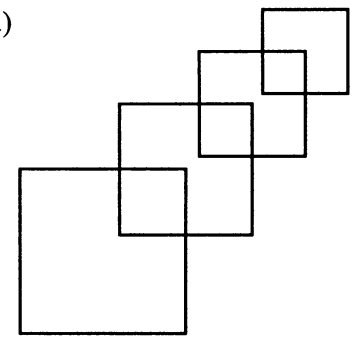
14. Să se modeleze mişcarea unui punct:
a) pe un pătrat; b) pe un triunghi.

15. Să se modeleze rotaţia unui segment în jurul mijlocului său.

B

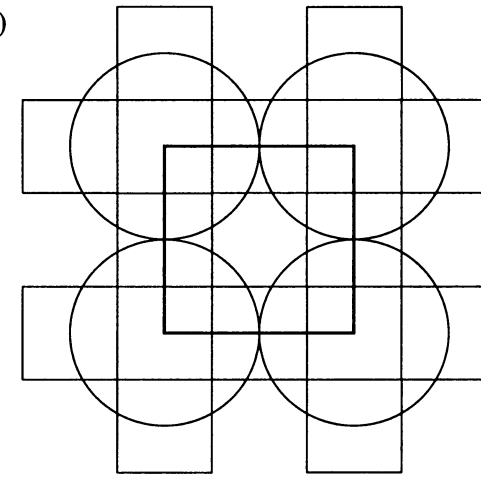
16. Să se realizeze un algoritm care, utilizînd un subprogram recursiv, va construi figuri ca în desen.

a)



Indicaţie
$l' = 0{,}8 \cdot l$

b)



Indicaţie
$r = \frac{1}{2} l_{\text{pătr}}$; $l'_{\text{pătr}} = r$ (l' este latura pătratului mai mic)

17. Să se explice ce se va afişa la ecran în urma execuţiei programului:
a)

```
program Graph6;
uses Crt,Graph;
const p=20;
var r,x,y:integer;
```

```
procedure initiere;
var  gd,gm:integer;
     CodEroare:integer;
begin
  gd:=Detect;
  InitGraph(gd,gm,'');
  CodEroare:=GraphResult;
  if CodEroare<>grOk then
     writeln('Graphics error:',GraphErrorMsg(CodEroare));
end;
procedure cerc(x,y,r:integer);
begin
  if r>p then begin
     cerc(x+r,y,r div 2);
     cerc(x,y+r,r div 2);
     cerc(x-r,y,r div 2);
     cerc(x,y-r,r div 2);
  end;
  circle(x,y,r);
end;
BEGIN
  initiere;
  cerc(GetMaxX div 2,GetMaxY div 2,100);
  readkey;
  closegraph;
END.
```

b)

```
program Graph7;
uses Crt,Graph;
const p=10;
var r,x,y:integer;
procedure initiere;
var  gd,gm:integer;
     CodEroare:integer;
begin
  gd:=Detect;
  InitGraph(gd,gm,'');
  CodEroare:=GraphResult;
  if CodEroare<>grOk then
     writeln('Graphics error:',GraphErrorMsg(CodEroare));
end;
procedure patrat(x,y,r:integer); forward;
procedure cerc(x,y,r:integer);
begin
  if r>p then begin
     patrat(x,y-r,r div 3);
     patrat(x+r,y,r div 3);
     patrat(x,y+r,r div 3);
     patrat(x-r,y,r div 3);
  end;
  circle(x,y,r);
end;
```

```
procedure patrat;
begin
  if r>p then begin
     cerc(x+r,y,r div 2);
     cerc(x,y+r,r div 2);
     cerc(x-r,y,r div 2);
     cerc(x,y-r,r div 2);
  end;
  rectangle(x-r,y-r,x+r,y+r);
end;
BEGIN
  initiere;
  cerc(GetMaxX div 2,GetMaxY div 2,150);
  readkey;
  closegraph;
END.
```

18. Se dau numerele reale a, b, c, x_0. Să se construiască graficul funcției $f: \mathbb{R} \to \mathbb{R}$, $f(x) = ax^2 + bx + c$, și tangenta la graficul funcției f în punctul x_0.

19. Se citește de la tastatură una din literele T, C, D, P.
- Dacă a fost citită litera T, să se deseneze un triunghi.
- Dacă a fost citită litera C, să se deseneze un cerc.
- Dacă a fost citită litera D, să se deseneze un dreptunghi.
- Dacă a fost citită litera P, să se deseneze un paralelogram.

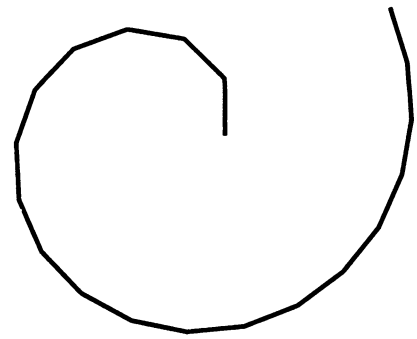
20. Să se deseneze un cerc fără a utiliza procedura `Circle`.

21. Să se deseneze o elipsă fără a utiliza procedura `Elipse`.

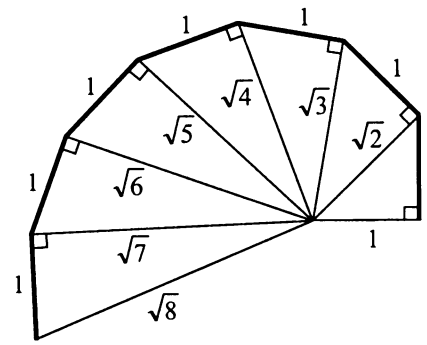
Indicație. Ecuația elipsei este $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$.

22. Să se modeleze mișcarea unei ghiulele de tun. Să se deseneze și tunul.

23. Să se deseneze „spirala lui Arhimede".



Indicaţie. Spirala se construieşte astfel:



24. Să se scrie o procedură care va desena un poligon regulat cu n laturi, unde n este parametrul procedurii.

25. Să se elaboreze un algoritm care va desena un pătrat. Utilizatorul va mişca pătratul în sus, în jos, în stînga, în dreapta cu ajutorul tastelor-săgeţi. Ieşirea din program se va realiza după apăsarea tastei ESC.
Indicaţie. Tastele-săgeţi au codurile: sus – 72, jos – 80, stînga – 75, dreapta – 77.

26. Să se elaboreze un algoritm care va desena curba definită parametric de ecuaţiile:

a) $x = \dfrac{3at}{1+t^3}$, $y = \dfrac{3at^2}{1+t^3}$ – *foiul lui Descartes*[1];

b) $x = \dfrac{at^2}{1+t^3}$, $y = \dfrac{at^3}{1+t^2}$ – *cisoida lui Diocles*[2];

c) $x = a(t - \sin t)$, $y = (1 - \cos t)$ – *cicloida*;

d) $x = a(\cos t)^3$, $y = a(\sin t)^3$ – *hipercicloida (astroida)*;

e) $x = a(\cos t + t \cdot \sin t)$, $y = a(\sin t - t \cdot \cos t)$ – *evolventa (desfăşurata) cercului*;

f) $x = 2a \cdot \cos t - a \cdot \cos 2t$, $y = 2a \cdot \sin t - a \cdot \sin 2t$ – *cardioida*;

h) $x = a \cdot (\cos t)^2 + b \cdot \cos t$, $y = a \cdot \cos t \cdot \sin t$, $a > 0$, $b > 0$, $0 \le t < 2\pi$.

27. Să se elaboreze un algoritm care va desena curba, a cărei ecuaţie este dată în coordonate polare:

a) $\rho = a\cos\varphi + b$, unde $a, b \in \mathbb{R}$, iar $\varphi \in [0, 2\pi]$ – *Melcul lui Pascal*[3];

b) $\rho^2 = 2a^2 \cos 2\varphi$, unde $a \in \mathbb{R}$, iar $\varphi = \left(-\dfrac{\pi}{4}, \dfrac{\pi}{4}\right) \cup \left(-\dfrac{\pi}{4}, \dfrac{5\pi}{4}\right)$ – *Lemniscata lui Bernoulli*[4];

---

[1] René Descartes (1596–1650) – matematician şi fizician francez.

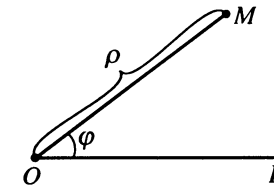
[2] Diocles (sec. II î. Hr.) – geometru grec.

[3] Blaise Pascal (1612–1662) – matematician şi fizician francez.

[4] Jacques Bernoulli (1654–1705) – matematician olandez.

c) $\rho = -\frac{a\cos 2\varphi}{\cos\varphi}$, unde $a \in \mathbb{R}$, $\varphi = [0, 2\pi]$ – *strofoida*.

Indicație. Coordonatele polare ρ și φ ale punctului M în plan semnifică distanța ρ de la punctul O (originea sau polul sistemului polar) la punctul M, iar φ reprezintă măsură unghiului MOP, unde [OP este axa polară a sistemului.



28. Să se elaboreze un algoritm care va permite utilizatorului să modifice poziția unui cerc cu ajutorul tastelor-săgeți și dimensiunile cercului cu ajutorul tastelor + și –. Cercul inițial are raza de 50 pixeli și centrul (200, 200).
Indicație. Tastele-săgeți au codurile: sus – 72, jos – 80, stînga – 75, dreapta – 77.

29. Să se elaboreze un algoritm care va permite utilizatorului să modifice dimensiunile unui dreptunghi cu ajutorul tastelor-săgeți. Dreptunghiul inițial are coordonatele unei diagonale (100, 100) și (300, 200).

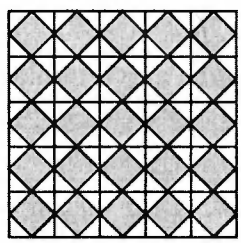
30. Să se modeleze rotația unui pătrat în jurul centrului său (în planul pătratului).

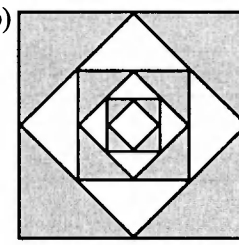
31. Să se modeleze rotația unui triunghi în jurul unuia dintre vîrfurile lui (în planul triunghiului).

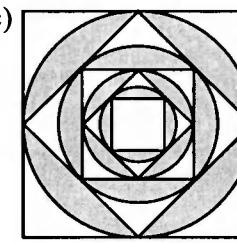
32. Să se modeleze mișcarea unei bile de biliard pînă cînd va fi apăsată tasta ESC. Ecranul va avea rolul mesei.

C

33. Să se deseneze imaginea:

a) 

b) 

c) 

34. Să se deseneze 30 de pătrate P_1, P_2, ..., P_{30}, fiecare înscris în precedentul, astfel încît vîrfurile pătratului P_{i+1} (i = 1, 2, ..., 29) să împartă fiecare latură a pătratului P_i (i = 1, 2, ..., 29) în raportul 0,08.

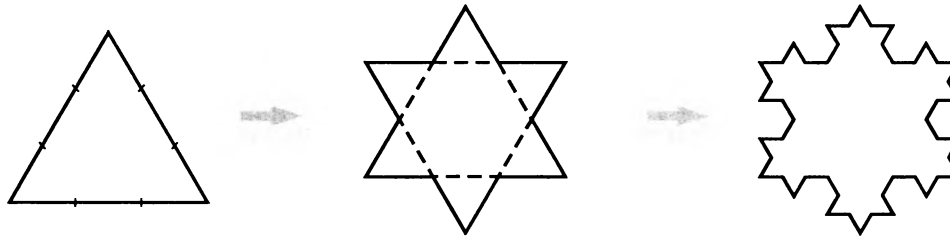
35. Se citește de la tastatură o cifră. Să se deseneze această cifră pe tot ectanul.

36. Să se deseneze un „panou electronic" format din 10×10 cercuri. Se citește de la tastatură o cifră. Se colorează interiorul cercurilor, astfel încît pe „panou" se obține cifra citită. Programul se execută pînă cînd se tastează un simbol care nu este cifră.

37. Poligonul ③ este construit din triunghiul echilateral ① astfel:

1. Fiecare latură se împarte în trei părți egale și pe ea se construiește un triunghi echilateral.

2. Se șterge baza triunghiurilor formate și se repetă procedeul cu fiecare latură a poligonului obținut.



①　　②　　③

Se dă numărul natural n. Să se construiască poligonul cu $3 \cdot 4^n$ laturi. De exemplu: pentru $n = 0$ se va construi un triunghi echilateral; pentru $n = 1$ se va construi poligonul ②, pentru $n = 2$ se va construi poligonul ③, pentru $n = 3$ se va construi un poligon cu 192 de laturi obținut din ③ repetînd pasul **2**.

38. Să se modeleze mișcarea unui inel în jurul unuia dintre diametrele sale.

16 Programarea orientată pe obiecte

Sugestii teoretice

Obiectul este un tip special de date şi metode (subprograme) pentru prelucrarea acestor date. Acţiunea de unire într-un tot întreg a datelor şi metodelor se numeşte **incapsulare**.

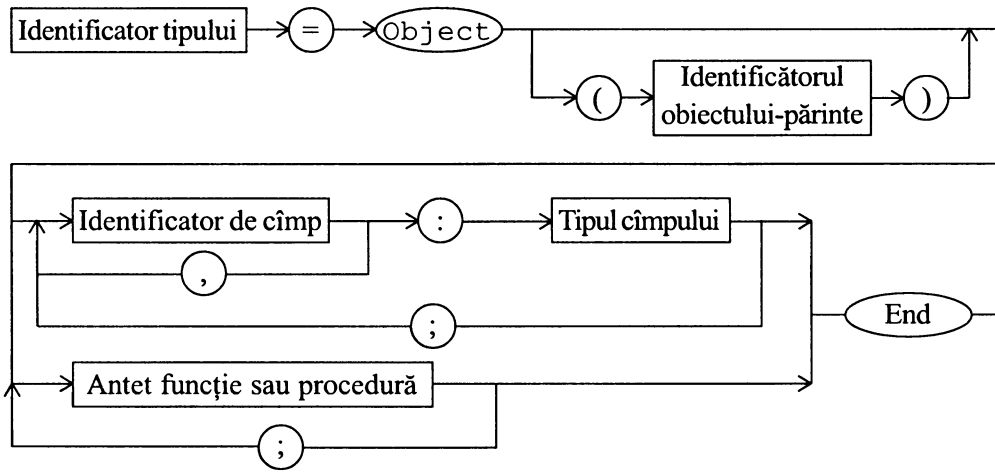
La declararea obiectului, în primul rînd se declară datele, apoi metodele lui.

Aceste metode doar se declară, indicîndu-se numai antetul lor. Descrierea metodelor se realizează mai jos, în textul programului.

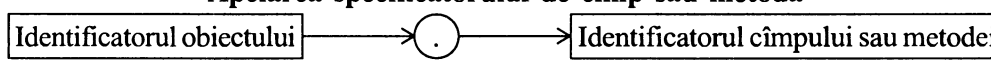
Moştenirea este o proprietate a obiectelor (moştenitorii) create în baza altor obiecte (strămoşi) de a păstra datele şi metodele strămoşilor.

Polimorfismul este o proprietate a obiectelor moştenitoare de a modifica structura metodelor părinţilor.

Declararea obiectului



Apelarea specificatorului de cîmp sau metodă



Observaţie

Ca şi în cazul înregistrărilor, identificatorii cîmpurilor şi metodelor pot fi apelaţi fără a include identificatorul obiectului, dacă se foloseşte cuvîntul-cheie **`with`**.

Metode statice şi virtuale. Obiecte dinamice

Corespondenţa dintre datele şi metodele obiectului realizată pînă la execuţia programului se numeşte **corespondenţă statică**, iar metodele respective – **metode statice**.

Corespondenţa realizată în timpul execuţiei programului se numeşte **corespondenţă dinamică**, iar metodele respective – **metode virtuale**.

Metodele virtuale sînt însoţite de directiva **`virtual`**.

Constructorul este o metodă specială (în loc de cuvîntul-cheie `procedure` se scrie cuvîntul-cheie `constructor`), care realizează legătura dintre obiect şi tabela metodelor virtuale (TMV) corespunzătoare lui.

Variabilele-obiect dinamice se creează ca şi variabilele dinamice obişnuite.

Pentru eliberarea memoriei ocupată de variabilele-obiecte dinamice se utilizează o metodă specială, numită **destructor** (în loc de cuvîntul-cheie `procedure` se scrie cuvîntul-cheie `destructor`). Spre deosebire de constructor, destructorul poate fi virtual.

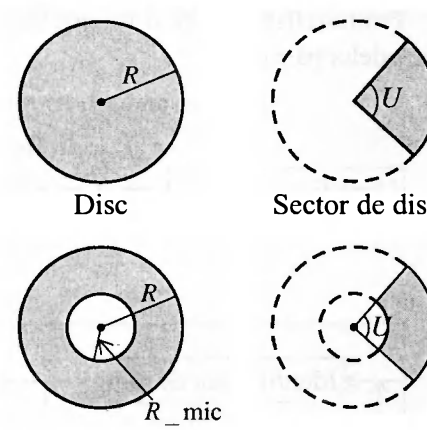
Probleme rezolvate

❶ Utilizînd tipul *obiect* să se descrie figurile geometrice Discul şi Coroana cu metode pentru calcularea:

- ariei, lungimii discului;
- ariei, lungimii exterioare, lungimii interioare ale coroanei circulare;
- ariei sectorului de disc şi ariei sectorului de coroană circulară.

De exemplu, pentru un disc cu raza de 10 cm şi o coroană circulară cu razele de 10 cm şi 2 cm vom avea

$\mathcal{A}_{disc} = \pi R^2 = 100\pi$ cm^2,
$\mathcal{P}_{disc} = 2\pi R = 20\pi$ cm,
$\mathcal{A}_{coroană} = \pi[R^2 - (R_mic)^2] = 96\pi$ cm^2,
$\mathcal{P}_{ext_coroana} = 2\pi R = 20\pi$ cm,
$\mathcal{P}_{int_coroana} = 2\pi \cdot R_mic = 4\pi$ cm.



Disc — Sector de disc

Coroană (circulară) — Sector de coroană (circulară)

Rezolvare:

```
program Obiectel;
uses Crt;
type  TDisc=object
        R:real;
        function Aria:real;virtual;
        function Lun:real;
        function Sector(u:integer):real;{Aria sectorului de masura u}
        constructor Init;
      end;
```

```
      TCoroana=object(TDisc) {mostenitor al obiectului TDisc}
         R_mic:real;
         function Aria:real;virtual;
         {modificam metoda Aria mostenita de la TDisc}
         function Lun_int:real;
         {Lungimea exterioara se va calcula cu metoda Lun a parintelui}
         constructor Init;
      end;
function TDisc.Aria;
begin
   Aria:=Pi*sqr(R);
   {datele obiectului sint globale pentru metodele lui}
end;
function TDisc.Lun;
begin
   Lun:=2*Pi*R;
end;
function TDisc.Sector;
begin
   Sector:=Aria/360*u;
end;
function TCoroana.Aria;
{modificam metoda parintelui - proprietatea polimorfism}
begin
   Aria:=Pi*(sqr(R)-sqr(R_mic));
end;
function TCoroana.Lun_int;
begin
   Lun_int:=2*Pi*R_mic;
end;
constructor TDisc.Init;
begin
end;
constructor TCoroana.Init;
begin
end;
var  Disc:TDisc;
     Coroana:TCoroana;
BEGIN
   ClrScr;
   Disc.Init;
   Coroana.Init;
   with Disc do begin
      write('Introdu raza discului: '); readln(R);
      writeln('Aria discului: ', Aria:2:2);
      writeln('Lungimea discului: ', Lun:2:2);
      writeln('Aria semidiscului: ', Sector(180):2:2);
   end;
   writeln('----------------------');
   with Coroana do begin
      write('Introdu raza mare a coroanei: '); readln(R);
      write('Introdu raza mica a coroanei: '); readln(R_mic);
      writeln('Aria coroanei: ',Aria:2:2);
```

```
      writeln('Lungimea exterioara a coroanei: ',Lun:2:2);
      {s-a apelat metoda parintelui}
      writeln('Lungimea interioara a coroanei: ',Lun_int:2:2);
      writeln('Aria semicoroanei: ',Sector(180):2:2);
   end;
   readkey;
END.
```

Observaţie

Modificaţi programul eliminînd constructorii şi declarînd metodele `Aria` statice (fără cuvîntul-cheie `virtual`).

Observaţi rezultatele metodelor Sector. Trageţi concluzia.

❷ (Фаронов В. В. [15] p. 182–194) Să se creeze o aplicaţie care va afişa la ecran un cerc, un dreptunghi şi un segment. Utilizatorul va putea deplasa cu ajutorul tastelor fiecare dintre figurile menţionate. Alegerea figurii se va realiza prin intermediul tastei Tab. De asemenea, la ecran vor apărea aleator 200 de puncte de culori aleatoare.

Rezolvare:

Aplicaţia va fi formată din 3 fişiere: primul fişier va fi un unit şi va descrie obiectele grafice menţionate, al doilea – un unit care va descrie obiectul-aplicaţie, iar al treilea va fi fişierul executabil care va crea, executa, apoi distruge un exemplar al aplicaţiei. O astfel de organizare a aplicaţiei este transparentă şi este caracteristică programării orientate pe obiecte.

```
unit GraphObj; {fisierul graphobj.pas}
INTERFACE
type  TGraphObj=object
        Private
          X,Y:Integer; {Coordonatele punctului de reper}
          Color:Word; {Culoarea figurii}
        Public
          Constructor Init(aX,aY:Integer; aColor:Word);
          {Creeaza un exemplar al obiectului}
          Procedure Draw(aColor:Word); Virtual;
          {Deseneaza un obiect cu culoarea aColor}
          Procedure Show; {Afiseaza obiectul}
          Procedure Hide; {Ascunde obiectul}
          Procedure MoveTo(dX, dY:Integer);
          {Deplaseaza obiectul in punctul X+dX, Y+dY}
      end;{TGraphObj}
      TPoint=object(TGraphObj)
        Procedure Draw(aColor:Word); Virtual;
      end;
      TLine=object(TGraphObj)
        dX,dY:Integer; {Cresterile coordonatelor extremitatii a II-a}
        Constructor Init(X1,Y1,X2,Y2:Integer; aColor:Word);
        Procedure Draw(aColor:Word); Virtual;
      end;
```

```
    TCircle=object(TGraphObj)
      R:integer;
      Constructor Init(aX,aY,aR:Integer; aColor:Word);
      Procedure Draw(aColor:Word); Virtual;
    end;
    TRect=object(TLine)
      Procedure Draw(aColor:Word); Virtual;
    end;
IMPLEMENTATION
  Uses Graph;
  Constructor TGraphObj.Init;
  begin
    X:=aX;
    Y:=aY;
    Color:=aColor
  end;
  Procedure TGraphObj.Draw;
  begin
    {In obiectul-parinte ea nu face nimic}
  end;
  Procedure TGraphObj.Show;
  begin
    Draw(Color)
  end;
  Procedure TGraphObj.Hide;
  begin
    Draw(GetBkColor)
  end;
  Procedure TGraphObj.MoveTo;
  begin
    Hide;
    X:=X+dX;
    Y:=Y+dY;
  end;
  {-Descriem metodele obiectelor mostenitoare-}
  Procedure TPoint.Draw; {Acopera metoda Draw a obiectului-parinte}
  begin
    PutPixel(X,Y,Color);
  end;
  Constructor TLine.Init;
  begin
    Inherited Init(X1,Y1,aColor); {apeleaza o metoda mostenita}
    dX:=X2-X1; {X1,Y1,X2,Y2 le are in antetul lui Init}
    dY:=Y2-Y1;
  end;
  Procedure TLine.Draw;
  begin
    SetColor(aColor);
    Line(X,Y,X+Dx,Y+dY)
  end;
  Constructor TCircle.Init;
  begin
    Inherited Init(aX,aY, aColor); {apeleaza o metoda mostenita}
```

```
      R:=aR
   end;
   Procedure TCircle.Draw;
   begin
      SetColor(aColor);
      Circle(X,Y,R)
   end;
   Procedure TRect.Draw;
   begin
      SetColor(aColor);
      Rectangle(X,Y,X+dX,Y+dY)
   end;
END.
```

Observaţie

Salvăm unit-ul GraphObj, apoi îl compilăm.

```
Unit GraphApp; {fisierul graphapp.pas}
INTERFACE
   Uses GraphObj;
   const NPoints=200;
   type  TGraphApp=object
         Points:array[1..NPoints] of TPoint;
         Line:Tline;
         Rect:TRect;
         Circ:TCircle;
         ActiveObj:Integer;
         Procedure Init;
         Procedure Run;
         Destructor Done;
         Procedure ShowAll;
         Procedure MoveActiveObj(dX,dY:Integer);
      end;
IMPLEMENTATION
   Uses Graph, Crt;
   Procedure TGraphApp.Init;
   var D,R,Err,k:Integer;
   begin
      D:=detect;
      InitGraph(D,R,'');
      Err:=GraphResult;
      if Err<>0 then begin
         GraphErrorMsg(Err);
         Halt
      end;
      Randomize;
      for k:=1 to NPoints do
         Points[k].Init(Random(GetMaxX),Random(GetMaxY),Random(15)+1);
      Line.Init(90,90,210,210,lightRed);
      Circ.Init(150,150,70,White);
      Rect.Init(100,100,200,200,Yellow);
      ShowAll;
```

```
      ActiveObj:=1
   end; {TGraphApp.Init;}
   Procedure TGraphApp.Run;
   var Stop:boolean;
   const D=5;
   begin
      Stop:=False;
      repeat
         case Readkey of
            #27:Stop:=True;
            #9:begin
               inc(ActiveObj);
               if ActiveObj>3 then ActiveObj:=1;
            end;
            #0: case Readkey of
               #71:MoveActiveObj(-D,-D); {stinga-sus}
               #72:MoveActiveObj(0,-D); {sus}
               #73:MoveActiveObj(D,-D); {dreapta-sus}
               #75:MoveActiveObj(-D,0); {stinga}
               #77:MoveActiveObj(D,0); {dreapta}
               #79:MoveActiveObj(-D,D); {stinga-jos}
               #80:MoveActiveObj(0,D); {jos}
               #81:MoveActiveObj(D,D); {dreapta-jos}
            end
         end;
         ShowAll;
      Until Stop
   end;
   Destructor TGraphApp.Done;
   begin
      CloseGraph;
   end;
   Procedure TGraphApp.ShowAll;
   var k:integer;
   begin
      for k:=1 to NPoints do Points[k].Show;
      Line.Show;
      Rect.Show;
      Circ.Show;
   end;
   Procedure TGraphApp.MoveActiveObj;
   begin
      case ActiveObj of
         1:Rect.MoveTo(dX,dY);
         2:Circ.MoveTo(dX,dY);
         3:Line.MoveTo(dX,dY);
      end;
   end;
END.
```

Observaţie

Salvăm unit-ul GraphApp, apoi îl compilăm.

```
Program Graph_Objects;
Uses GraphApp;
var App:TGraphApp;
BEGIN
   App.Init;
   App.Run;
   App.Done;
END.
```

Probleme propuse

A

1. Să se descrie obiectul „pătrat", al cărui cîmp va fi lungimea laturii pătratului. Obiectul va conține metode pentru calcularea perimetrului, ariei, lungimii diagonalei pătratului.

2. Să se descrie obiectul „bila", al cărui cîmp va fi raza bilei. Obiectul va conține metode pentru calcularea:
   - ariei suprafeței bilei;
   - volumului bilei.

3. Să se descrie obiectul „triunghi", ale cărui cîmpuri vor fi lungimile laturilor triunghiului. Obiectul va conține metode pentru:
   - calcularea perimetrului triunghiului;
   - calcularea ariei triunghiului;
   - calcularea înălțimii triunghiului corespunzătoare laturii specificate;
   - calcularea lungimii medianei triunghiului corespunzătoare laturii specificate;
   - calcularea lungimii bisectoarei unghiului triunghiului opus laturii specificate;
   - calcularea razei cercului circumscris triunghiului;
   - calcularea razei cercului înscris în triunghi;
   - determinarea tipului triunghiului după laturi;
   - determinarea tipului triunghiului după unghiuri.

4. Să se descrie obiectul „paralelogram", ale cărui cîmpuri vor fi lungimile laturilor paralelogramului şi măsura în grade a unghiului dintre două laturi alăturate. Obiectul va conține metode pentru determinarea:
   - înălțimilor paralelogramului;
   - perimetrului paralelogramului;
   - ariei paralelogramului;
   - lungimilor diagonalelor paralelogramului;
   - tipului paralelogramului (oarecare, romb, dreptunghi, pătrat).

5. Să se descrie obiectul „vector în plan", ale cărui cîmpuri vor fi coordonatele extremității vectorului (punctul $O(0, 0)$ va fi originea vectorului). Obiectul va conține metode pentru:

- calcularea lungimii vectorului;
- coordonatele mijlocului vectorului.

6. Să se descrie obiectul „grupă de persoane", ale cărui cîmpuri vor fi numele, prenumele şi data naşterii fiecărei persoane. Obiectul va conţine metode pentru:
- determinarea persoanei cu vîrsta cea mai mică (respectiv cea mai mare);
- determinarea vîrstei medii a grupului;
- afişarea persoanelor născute în luna specificată (respectiv în anul specificat).

7. Să se descrie obiectul „şir de numere întregi", ale cărui cîmpuri vor fi dimensiunea şi componentele şirului. Obiectul va conţine metode pentru:
- ordonarea crescătoare a componentelor şirului;
- determinarea componentei maximale (respectiv minimale);
- verificarea dacă şirul este ordonat crescător (respectiv descrescător);
- determinarea numărului de componente pozitive, negative, nule.

8. Să se descrie obiectul „student", ale cărui cîmpuri vor fi numele, prenumele, data naşterii (zi, lună, an), înălţimea, greutatea, anul înmatriculării, 3 note la ultima sesiune. Obiectul va conţine metode pentru:
- determinarea vîrstei în ani împliniţi (respectiv luni împlinite, zile);
- determinarea anului de studiu (considerînd că a fost înmatriculat la anul I);
- determinarea notei medii;
- determinarea dacă are sau nu bursă, considerînd că bursa se dă pentru nota medie mai mare decît 7.

B

9. Să se descrie obiectul „matrice de numere întregi", care va conţine metode pentru:
- determinarea numărului de componente pozitive, negative, nule;
- afişarea la ecran a liniei (coloanei) specificate;
- afişarea componentei maximale (minimale) şi a tuturor poziţiilor ei;
- determinarea rangului matricei.

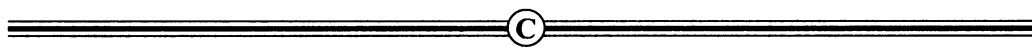
10. a) Să se descrie obiectul „monom de mai multe nedeterminate", ale cărui cîmpuri vor fi coeficientul şi partea literală a monomului. Obiectul va conţine metode pentru:
- calcularea gradului monomului;
- calcularea valorii monomului;
- ordonarea lexicografică a părţii literale.

b) Să se descrie obiectul „polinom de mai multe nedeterminate" – „copil" al obiectului descris în a). Obiectul va conţine metode pentru:
- calcularea gradului polinomului;
- calcularea valorii polinomului;
- aducerea la forma canonică a polinomului.

11. Să se descrie obiectul „agendă de telefoane" cu metode pentru:
- afişarea listei abonaţilor, al căror număr de telefon începe cu cifra dată;
- afişarea persoanei care are numărul de telefon dat;
- afişarea numărului de telefon a persoanei date.

12. a) Să se descrie obiectul „număr natural" cu metode pentru:
- determinarea cifrei de pe poziţia dată;
- determinarea numărului de cifre ale numărului;
- determinarea parităţii numărului;
- verificarea dacă cifra dată există în scrierea numărului.

b) Să se descrie obiectul „pereche de numere naturale" – „copil" al obiectului descris în a). Obiectul va conţine metode pentru:
- compararea numerelor perechii;
- determinarea cifrelor comune ambelor numere;
- verificarea dacă cifra dată există în scrierea ambelor numere;
- determinarea divizorilor comuni ai numerelor perechii.



13. Fie un sistem de axe ortogonale.

a) Să se descrie obiectul „punct", ale cărui cîmpuri vor fi coordonatele punctului. Obiectul va conţine o metodă pentru determinarea cadranului în care este situat punctul.

b) Să se descrie obiectul „segment" – „copil" al obiectului descris în a). Obiectul va conţine metode pentru:
- determinarea lungimii segmentului;
- determinarea coordonatelor mijlocului segmentului;
- verificarea dacă punctul dat aparţine segmentului, suportului segmentului sau nu este coliniar cu extremităţile lui.

c) Să se descrie obiectul „cerc" – „copil" al obiectului descris în a). Obiectul va conţine metode pentru:
- determinarea lungimii cercului;
- determinarea ariei discului;
- determinarea lungimii arcului de cerc de măsura dată;
- verificarea dacă punctul dat aparţine cercului, interiorului sau exteriorului cercului.

d) Să se descrie obiectul „dreptunghi" – „copil" al obiectului descris în b). Obiectul va conţine metode pentru:
- determinarea perimetrului dreptunghiului;
- determinarea ariei dreptunghiului;
- determinarea tipului dreptunghiului;
- determinarea coordonatelor centrului de simetrie al dreptunghiului.
- verificarea faptului că punctul dat aparţine dreptunghiului, interiorului sau exteriorului dreptunghiului.

e*) Să se descrie obiectul „triunghi"– „copil" al obiectului descris în b). Obiectul va conține metode pentru:

- determinarea perimetrului triunghiului;
- determinarea ariei triunghiului;
- determinarea lungimilor medianelor, bisectoarelor, înălțimilor triunghiului;
- determinarea tipului triunghiului după laturi (scalen, isoscel, echilateral);
- determinarea tipului triunghiului după unghiuri (ascuțitunghic, dreptunghic, optuzunghic);
- determinarea razei și coordonatelor centrului cercului circumscris triunghiului (cercului înscris în triunghi);
- determinarea coordonatelor punctelor de tangență a triunghiului cu cercul înscris în triunghi;
- verificarea dacă punctul dat aparține triunghiului, interiorului sau exteriorului triunghiului.

f*) Se dau un punct, un segment, un cerc, un dreptunghi și un triunghi (cîte un exemplar al obiectelor descrise în a)-e*)). Să se determine:

- poziția punctului față de fiecare dintre celelalte figuri geometrice;
- poziția dreptei suport a segmentului față de cerc (tangentă, secantă, exterioară);
- poziția cercului față de dreptunghi (intersectează, aparține interiorului, exteriorului etc.);
- figura cu aria cea mai mare (respectiv cea mai mică);
- distanța de la punct la segment (respectiv cerc, dreptunghi, triunghi).

17 Grafuri (neorientate)

Sugestii teoretice

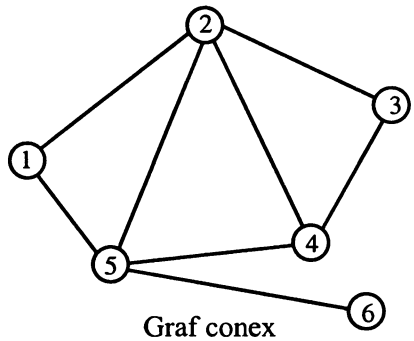
Un **graf** (neorientat) este o pereche ordonată $G = (V, M)$, unde V este o mulţime finită şi nevidă de elemente, numite **vîrfuri** (sau **noduri**), iar $M \subseteq V \times V$ este o mulţime de perechi neordonate, numite **muchii**.

Vîrfurile i, j ale oricărei muchii $m = [i, j]$ se numesc **vîrfuri adiacente**. Muchia m se numeşte **incidentă** cu vîrful i şi cu vîrful j. Vîrfurile i şi j se numesc **extremităţile** muchiei m.

Gradul unui vîrf v (se notează $d(v)$) este numărul de muchii incidente cu v. **Gradul grafului** este suma gradelor vîrfurilor lui. Dacă $d(v) = 0$, atunci v se numeşte **vîrf izolat**. Dacă $d(v) = 1$, atunci v se numeşte **vîrf terminal**.

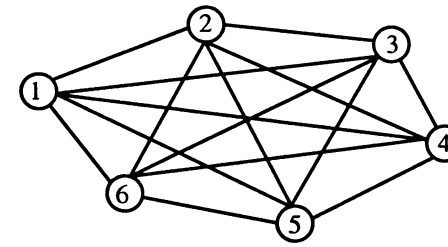
Un graf se **reprezintă grafic** în plan astfel: fiecare vîrf se reprezintă printr-un punct (sau un cerc), iar fiecare muchie – printr-un segment ale cărui extremităţi vor fi punctele corespunzătoare extremităţilor muchiei. De regulă, mulţimea V a vîrfurilor unui graf cu n vîrfuri se consideră mulţimea $\{1, 2, \ldots, n\}$.

În figura 1a) este reprezentat graful $G = (V, M)$, unde $V = \{ 1, 2, \ldots, 6\}$, iar $M = \{ [1, 2], [1, 5], [2, 3], [2, 4], [2, 5], [3, 4], [4, 5], [5, 6] \}$.



Graf conex

a)



Graf complet

b)

Fig. 1

Un graf se numeşte **graf complet** dacă fiecare două vîrfuri ale acestuia sînt adiacente (figura 1b).

Graful $G_p = (V, M_1)$, unde $M_1 \subseteq M$, se numeşte **graf parţial** al grafului $G = (V, M)$.

Graful $G_s = (V_1, M_1)$, unde $V_1 \subseteq V$, iar M_1 conține toate muchiile din M care au extremitățile în V_1, se numește **subgraf** al grafului $G = (V, M)$.

În figura 1a) graful $G_1 = (V, M_1)$, unde $M_1 = \{ [1, 2], [2, 3], [2, 4], [2, 5] \}$, este graf parțial al grafului G, iar graful $G_2 = (V_1, M_2)$, unde $V_1 = \{4, 5, 6\}$ și $M_2 = \{[4, 5], [5, 6]\}$, este subgraf al grafului G.

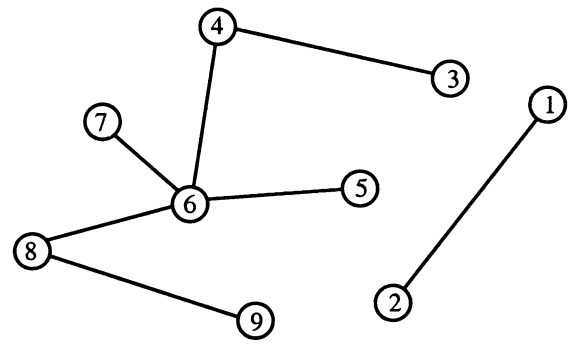
Fie graful $G = (V, M)$ cu n vîrfuri. Succesiunea de vîrfuri $L = [i_1, i_2, \ldots, i_k]$, unde $i_s \in \{1, 2, ..., n\}$ $(s = \overline{1, k})$ se numește **lanț** dacă orice două vîrfuri consecutive din L sînt adiacente.

Lanțul $L = [i_1, i_2, \ldots, i_k]$ se numește **ciclu** dacă $i_1 = i_k$.

Lanțul (respectiv ciclul) se numește **elementar** dacă fiecare două vîrfuri ale lui (în cazul ciclului în afară de primul și ultimul) sînt diferite.

În figura 1a) succesiunile $L_1 = [1, 2, 3, 4, 5]$, $L_2 = [1, 2, 4, 5, 1]$, $L_3 = [1, 5, 6]$, $L_4 = [1, 2, 3, 4, 5, 1]$, $L_5 = [3, 4, 2, 5, 4, 3]$ sînt lanțuri. Lanțurile L_2, L_4 și L_5 sînt cicluri. Ciclul L_4 este ciclu elementar, iar ciclul L_5 nu este ciclu elementar.

Un graf $G = (V, M)$ se numește **graf conex** dacă pentru orice două vîrfuri ale acestuia există un lanț care leagă aceste vîrfuri. Graful din figura 1a) este graf conex. Graful din figura 2 nu este conex.



Graf neconex

Fig. 2

Un ciclu (lanț) elementar al unui graf G care conține toate vîrfurile grafului se numește **ciclu (lanț) hamiltonian**. Graful G se numește **graf hamiltonian** dacă conține un ciclu hamiltonian.

Un ciclu al unui graf G care conține toate muchiile grafului se numește **ciclu eulerian**. Graful G se numește **graf eulerian** dacă conține un ciclu eulerian.

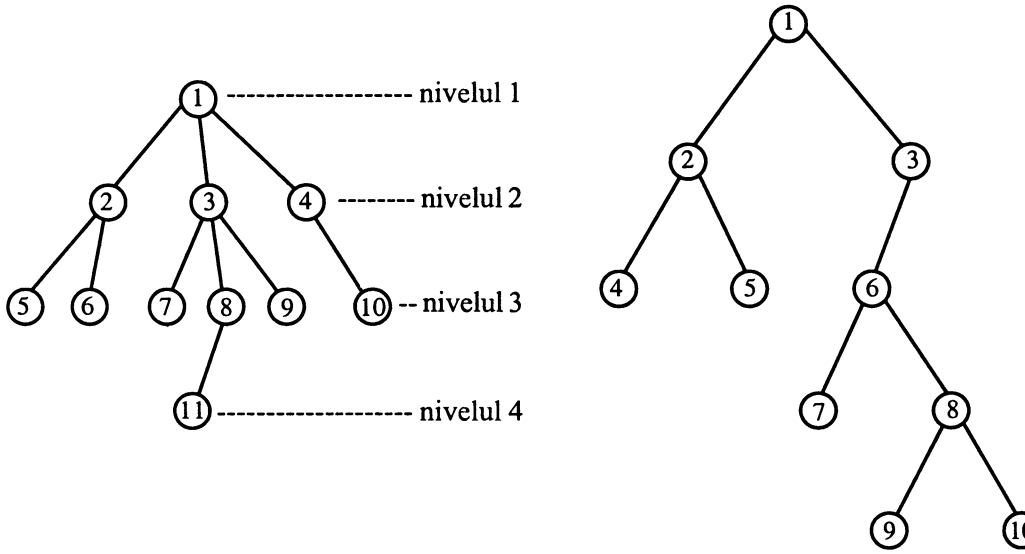
Un lanț elementar al unui graf G care conține toate muchiile grafului, fiecare o singură dată, se numește **lanț eulerian**.

Un graf conex și fără cicluri se numește **arbore** (fig. 3a).

Un **arbore binar** este un arbore care conține un nod special, numit **rădăcină**, iar celelalte noduri sînt repartizate în două mulțimi disjuncte și fiecare dintre aceste mulțimi este, la rîndul ei, un arbore (fig. 3b).

Vîrfurile arborelui adiacente cu rădăcina se numesc **descendenţii** rădăcinii.

Se consideră că rădăcina se află pe primul **nivel**, iar descendenţii ei – pe nivelul al doilea. Descendenţii fiecărui nod de pe nivelul doi se consideră noduri de pe nivelul 3 ş.a.m.d. Un nod este **părintele** altui nod dacă ultimul este descendentul primului. Descendenţii aceluiaşi nod se numesc **fraţi**.



arbore — a)

arbore binar — b)

Fig. 3

Un graf $G = (V, M)$ se poate **reprezenta în calculator** folosind diferite structuri de date:

a) cu ajutorul **matricei de adiacenţă** – o matrice pătrată de ordinul n formată din 0 şi 1, unde n este numărul de vîrfuri ale grafului şi

$$a_{ij} = \begin{cases} 1, \text{ dacă } [i, j] \in M \\ 0, \text{ dacă } [i, j] \notin M \end{cases} \text{ (evident } a_{ij} = a_{ji});$$

b) cu ajutorul **tipului de date articol**: definim tipul-articol *muchie*, apoi formăm un vector cu m elemente de tip *muchie*, unde m este numărul de muchii ale grafului.

```
Type muchie=record
              e1,e2:byte; {e1 si e2 sint extremitatile muchiei}
            end;
graf=array[1..m] of muchie;
```

(numărul n al vîrfurilor se consideră dat).

c) cu ajutorul a doi vectori e_1 şi e_2 cu m componente fiecare, astfel încît $e_1[i]$ este o extremitate a muchiei i, iar $e_2[i]$ – cealaltă extremitate a muchiei i (numărul n al vîrfurilor se consideră dat).

Un **arbore binar poate fi reprezentat în calculator**:

a) cu ajutorul a doi vectori *st* şi *dr* cu *n* componente fiecare, unde *n* este numărul de noduri ale arborelui binar, iar *st*[*i*], *dr*[*i*] sînt respectiv descendentul drept şi descendentul stîng al nodului *i* (rădăcina *r* se consideră dată). Valoarea 0 a unei componente semnifică faptul că descendentul respectiv lipseşte.

b) cu ajutorul a doi vectori *t* şi *d* cu *n* componente fiecare, unde *n* este numărul de noduri ale arborelui binar, iar *t*[*i*], *d*[*i*] sînt respectiv tatăl nodului *i* şi descendentul stîng (respectiv drept), egal cu –1 (respectiv egal cu 1), al tatălui *t*[*i*] (rădăcina *r* se consideră dată). Valoarea 0 a unei componente a vectorului *d*[*i*] semnifică faptul că descendentul respectiv lipseşte. Evident *t*[*r*] = 0, unde *r* este rădăcina arborelui.

c) Un arbore binar poate fi construit în calculator şi prin alocarea dinamică a memoriei. Astfel, ficecare nod, în afară de numărul nodului (în caz mai general fiecare nod al unei structuri arborescente poate conţine şi alte tipuri de informaţii), va conţine şi 2 pointeri: unul către subarborele stîng, altul către cel drept.

```
type p=^nod;
     nod=record
           i:byte,
           s,d:p;
         end;
```

Prin **parcurgerea grafului** (în particular a arborelui) se înţelege vizitarea tuturor vîrfurilor grafului, plecînd de la un vîrf dat *i*, astfel încît fiecare vîrf accesibil din *i* pe muchii adiacente două cîte două să fie atins o singură dată.

Un graf poate fi parcurs în două moduri:

a) Metoda **Breadth First** (parcurgerea în lăţime) – se vizitează vîrful dat *i*, apoi „vecinii lui", apoi „vecinii" nevizitaţi ai acestora ş.a.m.d.;

b) Metoda **Depth First** (parcurgerea în adîncime) – se vizitează vîrful dat *i*, apoi primul vecin, apoi primul vecin nevizitat al acestuia ş.a.m.d.; dacă după vizitatrea nodului *j* se vizitează un vecin, iar acesta nu are un vecin nevizitat, atunci se caută următorul vecin al lui *j* ş.a.m.d.

Deosebim trei moduri de **parcurgere a unui arbore binar**:

a) parcurgerea în **preordine** – se vizitează rădăcina, apoi subarborele stîng, apoi subarborele drept;

b) parcurgerea în **inordine** – se parcurge subarborele stîng, apoi rădăcina, apoi subarborele drept;

c) parcurgerea în **postordine** – se parcurge subarborele stîng, apoi subarborele drept, apoi rădăcina.

Probleme rezolvate

❶ Se dă un graf. Să se reprezinte graful în plan.

Rezolvare:

Pentru a evita suprapunerea nodurilor şi muchiilor, vom plasa vîrfurile circular (fiecare 3 puncte ale unui cerc sînt necoliniare).

```
program desen_graf;
uses crt,graph;
var e1,e2:array[1..50] of byte; {e1,e2 - definesc graful}
    x,y:array[1..50] of integer;{coordonatele virfurilor grafului}
    i,m,n:integer; {m - numarul de muchii, n - numarul de virfuri}
    alfa:real;
    v:string;
procedure initiere;
var gd,gm,ErrCode:integer;
begin
   gd:=Detect;
   InitGraph(gd,gm,'c:\tp\bgi');
   ErrCode:=GraphResult;
   if ErrCode<>grOk then
      write('Eroare grafica: ',GraphErrorMsg(ErrCode));
end;
BEGIN
   ClrScr;
   write('Scrie numarul de virfuri: ');
   readln(n);
   write('Scrie numarul de muchii: ');
   readln(m);
   writeln('Scrie muchiile: ');
   for i:=1 to m do begin
      write('Muchia ',i,': ');
      readln(e1[i],e2[i]);
   end;
   initiere;
   {construim virfurile grafului}
   alfa:=2*pi/n;
   for i:=1 to n do begin
      x[i]:=getmaxx div 2+round(200*cos(alfa*i));
      y[i]:=getmaxy div 2-round(200*sin(alfa*i));
      setcolor(5);
      circle(x[i],y[i],15);
      str(i,v);
      setcolor(15);
      outtextxy(x[i]-5,y[i],v);
   end;
   {construim muchiile grafului}
   setcolor(7);
   for i:=1 to m do line(x[e1[i]],y[e1[i]],x[e2[i]],y[e2[i]]);
   readkey;
   closegraph;
END.
```

❷ Se dă un graf prin matricea sa de adiacență. Să se parcurgă graful prin metoda Depth First (DF).

Rezolvare:

Vom folosi o procedură recursivă Pl_df(*Pl*), care parcurge prin metoda DF graful plecînd din vîrful *Pl*. Evident, vizitînd un vîrf *i*, în continuare vom parcurge celelale vîrfuri similar, adică vom executa Pl_df(*i*).

```
program DF;
uses Crt;
var a:array[1..20,1..20] of 0..1; {matrice de adiacenta a grafului}
    parcurs:array[1..20] of 0..1; {parcurs[i]=1 daca nodul i a fost
                                   parcurs}
    n,m,i,j,x,y:integer;
    c:array[1..20] of integer;
procedure Pl_df(pl:integer);
var j:integer;
begin
   write(pl:3);
   parcurs[pl]:=1;
   for j:=1 to n do
      if (a[pl,j]=1) and (parcurs[j]=0) then pl_df(j);
end;
BEGIN
   ClrScr;
   write('Scrie numarul de virfuri si muchii: ');
   readln(n,m);
   for i:=1 to n do
      for j:=1 to n do A[i,j]:=0;
   writeln('Scrie muchiile ');
   for i:=1 to m do begin
      write('Muchia ',i,': ');
      readln(x,y);
      A[x,y]:=1;A[y,x]:=1;
   end;
   write('Virful de plecare: ');
   readln(i);
   Pl_df(i);
   readkey;
END.
```

❸ Se dă un graf prin matricea sa de adiacenţă. Să se afişeze componentele conexe ale grafului. (Evident, un graf este conex dacă şi numai dacă are o singură componentă conexă.)

Rezolvare:

1. Se caută nodul nevizitat (acel nod *i* pentru care *parcurs*[*i*]=0).

2. Se porneşte de la nodul *i* şi se vizitează în adîncime (se execută Pl_df(i)) toate nodurile accesibile, adică nodurile pentru care există un lanţ care le leagă cu nodul *i*. În timpul vizitării marcăm nodurile. Dacă nodul *j* a fost vizitat, atunci *parcurs*[*j*]=1. Acestea formează cu vîrful *i* o componentă conexă.

3. Căutăm următorul nod nevizitat. Dacă există un astfel de nod, se trece la pasul **1** (se determină altă componentă conexă), în caz contrar este sfîrşit de algoritm.

```
program nr_conexe;
uses Crt;
var a:array[1..20,1..20] of 0..1; {matrice de adiacenta a grafului}
    parcurs:array[1..20] of 0..1; {parcurs[i]=1 daca nodul i a fost
                                   parcurs}
    n,m,i,j,x,y,nc:integer; {nc - numarul de componente conexe}
```

```
    c:array[1..20] of integer;
    este_nod:boolean;
procedure Pl_df(pl:integer);
var j:integer;
begin
   write(pl:3);
   parcurs[pl]:=1;
   for j:=1 to n do
   if (a[pl,j]=1) and (parcurs[j]=0) then pl_df(j);
end;
BEGIN
   ClrScr;
   write('Scrie numarul de virfuri si muchii: ');
   readln(n,m);
   for i:=1 to n do
     for j:=1 to n do A[i,j]:=0;
   writeln('Scrie muchiile ');
   for i:=1 to m do begin
     write('Muchia ',i,': ');
     readln(x,y);
     A[x,y]:=1; A[y,x]:=1;
   end;
   for j:=1 to n do parcurs[j]:=0;
   nc:=0;
   repeat
     este_nod:=false;
     i:=1;
     while (i<=n) and (not este_nod) do begin
           if parcurs[i]=0 then este_nod:=true;
           i:=i+1;
     end;
     if este_nod then begin
           inc(nc);
           write('Componenta conexa ',nc,' : ');
           pl_df(i-1);
           writeln;
     end;
   until not este_nod;
   if nc=1 then write('Graful este conex');
   readkey;
END.
```

❹ Se dau *n* oraşe. Dacă între două oraşe există drum direct, atunci acestea pot fi conectate telefonic direct. Să se conecteze oraşele într-o reţea telefonică, astfel încît fiecare două oraşe să fie conectate direct sau prin intermediul altora şi costul conectării să fie minim. Pentru fiecare două oraşe se ştie dacă există sau nu drum direct, precum şi costul conectării directe.

Rezolvare:

Problema poate fi modelată matematic astfel:

Se dă un graf cu *n* vîrfuri. Pentru fiecare muchie (drumul direct dintre două oraşe) se ştie costul muchiei. Trebuie să se determine o componentă conexă a grafului care va

conține toate vîrfurile grafului și suma costurilor muchiilor componentei va fi minimă.

De exemplu, pentru graful din figura 4 (pe fiecare muchie este scris costul ei) componenta conexă căutată este: $G_1 = (V, M_1)$, unde $M_1 = \{ [1, 4], [2, 6], [3, 4], [3, 6], [3, 5] \}$. Costul componentei este 9.

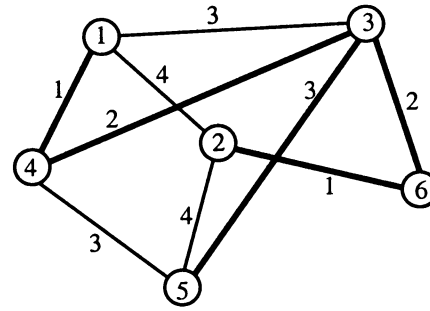


Fig. 4

Se poate ușor demonstra că soluția problemei este un arbore. De aceea problema enunțată se mai numește *problema construirii arborelui parțial de cost minim.*

Se cunosc mai mulți algoritmi de rezolvare a problemei menționate. Unul din ei este *Algoritmul lui Kruskal*[1] (publicat în 1956).

1. Considerăm inițial n arbori disjuncți (fiecare 2 arbori nu au muchii comune): T_1, T_2, ..., T_n, unde $T_i = (\{i\}, \varnothing)$, $i = \overline{1, n}$ (arborele T_i constă doar din vîrful i și nu are nici o muchie).

2. Dintre toate muchiile alegem muchia $[i, j]$ de cost minim.

3. Unim arborii T_i și T_j, astfel încît vom obține $n - 1$ arbori disjuncți. Renotînd, obținem arborii T_1, T_2, ..., T_{n-1}.

4. Dintre muchiile rămase alegem muchia $[i, j]$ de cost minim. Unim arborii T_i și T_j și obținem $n - 2$ arbori disjuncți. Renotînd, obținem arborii T_1, T_2, ..., T_{n-2}.

5. Dintre muchiile rămase alegem muchia $[i, j]$ de cost minim cu extremitățile aparținînd diferiților arbori (în caz contrar apare ciclu). Unim arborii T_i și T_j și obținem $n - 3$ arbori disjuncți. Renotînd, obținem arborii T_1, T_2, ..., T_{n-3} ș. a. m.d. pînă cînd obținem un arbore care conține toate vîrfurile grafului (evident, arborele va fi de cost minim). Vom efectua $n - 2$ uniri de arbori.

Pentru reprezentarea arborelui folosim metoda b) de reprezentare a grafurilor, însă în definiția tipului *muchie* mai adăugam cîmpul c al costului muchiei.

De asemenea, vom utiliza vectorul A de numere naturale pentru care $a_i = a_j$ dacă și numai dacă vîrfurile i și j aparțin aceluiași arbore T. Inițial $a_i = i$.

```
program Kruskal;
uses Crt;
type muchie=record
              e1,e2:byte;
```

[1] Joseph B. Kruskal (n. 1929) – matematician american.

```
            c:real;
          end;
var graf:array[1..30] of muchie;
    a:array[1..50] of byte;
    temp:muchie;
    n,m,i,j,v1,v2,k:byte;
    f:boolean;
    Cost:real;
BEGIN
   ClrScr;
   write('Introdu numarul de virfuri ale grafului: ');
   readln(n);
   write('Introdu numarul de muchii ale grafului: ');
   readln(m);
   for i:=1 to m do begin
      writeln('- Muchia ',i,' -');
      write('Extremitatile: ');
      readln(graf[i].e1,graf[i].e2);
      write('Costul: ');
      readln(graf[i].c);
   end;
   {ordonarea vectorului GRAF dupa costul muchiilor prin metoda bulelor}
   Cost:=0;
   for i:=1 to n do
      a[i]:=i;
   repeat
      f:=false;
      for i:=1 to m do
         if graf[i].c>graf[i+1].c then begin
            temp:=graf[i];
            graf[i]:=graf[i+1];
            graf[i+1]:=temp;
            f:=true;
         end;
   until not f;
   {sfirsit ordonare}
   k:=0;
   i:=1;
   writeln('----- Arborele de cost minim -----');
   while k<n-1 do begin
      if A[graf[i].e1]<>a[graf[i].e2] then begin
         inc(k);
         cost:=cost+graf[i].c;
         write('[',graf[i].e1,',',graf[i].e2,'] ');
         {afisarea muchiei i in ordinea cresterii costului}
         v1:=A[graf[i].e1];
         v2:=A[graf[i].e2];
         for j:=1 to n do
            if a[j]=v1 then a[j]:=v2;
      end;
      inc(i);
   end;
   writeln;
```

```
   write('Costul arborelui: ', cost:2:2);
   readkey;
END.
```

❺ Se dă un arbore. Să se elaboreze un algoritm care parcurge arborele prin metoda selectată de utilizator.

Rezolvare:

```
program parc_arbore;
uses crt;
var s,d:array[1..20] of byte; {vectorii S si D definesc descendentii}
    r,n,i,k:byte; {R este radacina, N este numarul de noduri}
    c:char;
procedure preordine(j:byte);
begin
   write(j:3);
   if s[j]<>0 then preordine(s[j]);
   if d[j]<>0 then preordine(d[j]);
end;
procedure inordine(j:byte);
begin
   if s[j]<>0 then inordine(s[j]);
   write(j:3);
   if d[j]<>0 then inordine(d[j]);
end;
procedure postordine(j:byte);
begin
   if s[j]<>0 then postordine(s[j]);
   if d[j]<>0 then postordine(d[j]);
   write(j:3);
end;
BEGIN
   ClrScr;
   write('Introdu numarul total de noduri: ');
   readln(n);
   write('Scrie radacina: ');
   readln(r);
   writeln('Pentru fiecare nod scrie descendentul sting, apoi cel drept.');
   writeln('Daca descendentul lipseste, scrie 0.');
   i:=1;
   k:=1;
   repeat
      write('Nodul ',i,': ');
      readln(s[i],d[i]);
      if s[i]<>0 then inc(k);
      if d[i]<>0 then inc(k);
      inc(i)
   until k=n;
   writeln('Alege modul de parcurgere: ');
   writeln('1 - preordine');
   writeln('2 - inordine');
   writeln('3 - postordine');
   repeat
      readln(c);
```

```
   until c in ['1'..'3'];
   case c of
      '1': preordine(r);
      '2': inordine(r);
      '3': postordine(r);
   end;
   readkey;
END.
```

❻ Se dă un arbore binar. Să se reprezinte arborele în plan.

Rezolvare:

```
program des_arbore;
uses crt,graph;
type coord=record
             x,y:integer;
          end;
var s,d:array[1..40] of byte; {vectorii S si D definesc subarborii}
    r,n,i,k:byte; {R este radacina, N este numarul de noduri}
    c:array[1..40] of coord;
    v:string;
procedure des_preord(j:byte);
begin
   circle(c[j].x,c[j].y,10);
   str(j,v);
   outtextxy(c[j].x-5,c[j].y-5,v);
   if s[j]<>0 then begin
      inc(k);
      c[s[j]].x:=c[j].x-100+k*15; {unghiul dintre descendenti la fiecare
                                   nivel se va micsora}
      c[s[j]].y:=c[j].y+40;
      line(c[j].x-10,c[j].y+10,c[s[j]].x+10,c[s[j]].y-10);
      preordine(s[j]);
      dec(k);
   end;
   if d[j]<>0 then begin
      inc(k);
      c[d[j]].x:=c[j].x+100-k*15;
      c[d[j]].y:=c[j].y+40;
      line(c[j].x+10,c[j].y+10,c[d[j]].x-10,c[d[j]].y-10);
      preordine(d[j]);
      dec(k);
   end;
end;
procedure initiere;
var gd,gm,ErrCode:Integer;
begin
   gd:=Detect;
   InitGraph(gd,gm,'c:\tp\bgi');
   ErrCode:=GraphResult;
   if ErrCode<>grOk then
      write('Eroare grafica: ',GraphErrorMsg(ErrCode));
end;
```

```
BEGIN
   ClrScr;
   write('Introdu numarul total de noduri: ');
   readln(n);
   write('Scrie radacina: ');
   readln(r);
   writeln('Pentru fiecare nod scrie nodul sting, apoi cel drept.');
   writeln('Daca nodul lipseste, scrie 0.');
   i:=1;
   k:=1;
   repeat
      write('Nodul ',i,': ');
      readln(s[i],d[i]);
      if s[i]<>0 then inc(k);
      if d[i]<>0 then inc(k);
      inc(i);
   until k=n;
   initiere;
   k:=0;
   c[r].x:=getmaxX div 2;
   c[r].y:=50;
   des_preord(r);
   readkey;
   closegraph;
END.
```

❼ Se dă un vector ale cărui componente sînt numere întregi. Să se ordoneze crescător vectorul utilizînd o structură arborescentă.

Rezolvare:

Vom construi arborele ordonat (numit şi **arbore binar de căutare**) astfel:

1. Iniţial rădăcina este a_1, iar descendenţii ei nu există (fiecare dintre pointerii spre ei are valoarea NIL).

2. Dacă a_i este mai mică decît nodul R (iniţial R este rădăcina), atunci a_i se va compara cu descendentul stîng al nodului R, altfel – cu cel drept. Dacă descendentul cu care urmează a fi realizată comparaţia lipseşte, atunci a_i devine acest descendent. Parcurgînd arborele obţinut prin metoda inordine, vom citi componentele vectorului în ordine crescătoare. De exemplu, pentru vectorul –1, 0, 5, 2, 3, 6, –2, 5 obţinem următorul arbore:

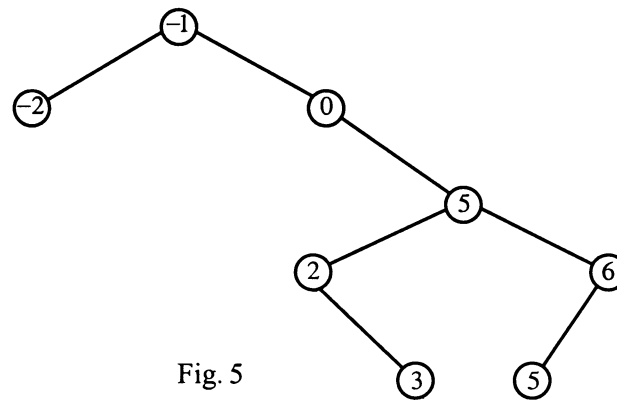


Fig. 5

```
program Sort_bin;
uses Crt;
type p=^nod;
     nod=record
           val:integer;
           s,d:p;
     end;
var  a:array[1..50] of integer;
     rad:p;
     n,i,k:integer;
procedure arb_ord(x:integer; var r:p); {R este nodul cu care urmeaza a
                                        fi realizata comparatia}
begin
   if r=nil then begin {daca nodul R lipseste, X devine acest nod}
      new(r);
      r^.val:=x;
      r^.s:=nil;
      r^.d:=nil;
   end else
      if x<r^.val then arb_ord(x,r^.s)
        else arb_ord(x,r^.d);
end;
procedure cit_arb(r:p; niv:integer); {Niv este nivelul nodului}
begin
   if r<>nil then begin
      cit_arb(r^.s,niv+1);
      a[k]:=r^.val;
      inc(k);
      cit_arb(r^.d,niv+1);
   end;
end;
BEGIN
   ClrScr;
   write('Introdu numarul de componente ale vectorului: ');
   readln(n);
   rad:=nil;
   for i:=1 to n do begin
      write('A[',i,']: ');
      readln(a[i]);
      arb_ord(a[i],rad);
   end;
   k:=1;
   cit_arb(rad,1); {incepem cu nivelul 1}
   for i:=1 to n do write(a[i]:3);
   readkey;
END.
```

Probleme propuse

A

1. Numărul total de grafuri cu n vîrfuri este $t = 2^{\frac{n(n-1)}{2}}$.
Se dă numărul natural n. Să se elaboreze un algoritm care va afişa la ecran cele t matrice de adiacenţă (corespunzătoare celor t grafuri).

2. Se dă un graf prin matricea sa de adiacenţă. Să se elaboreze un algoritm care va afişa la ecran:
a) vîrfurile terminale; b) vîrfurile izolate;
c) vîrfurile de grad par; d) vîrfurile de grad maxim.

3. Se dă un graf prin matricea sa de adiacenţă şi un vîrf v. Să se elaboreze un algoritm care va afişa la ecran muchiile (dacă există) incidente cu vîrful v.

4. Se dă un graf prin matricea sa de adiacenţă. Să se verifice dacă graful este complet.

5. Se dau matricele de adiacenţă ale grafurilor $G = (V, M)$ şi $G_1 = (V, M_1)$. Să se verifice dacă G_1 este un graf parţial al grafului G.

6. Se dau matricele de adiacenţă ale grafurilor $G = (V, M)$ şi $G_1 = (V_1, M_1)$, unde $V_1 \subseteq V$. Să se verifice dacă G_1 este un subgraf al grafului G.

7. Se dă un graf cu n vîrfuri prin matricea sa de adiacenţă şi vîrfurile $v_1, v_2, \ldots, v_s$, unde $s \leq n$. Să se verifice dacă vîrfurile în ordinea dată formează:
a) un lanţ; b) un ciclu;
c) un lanţ elementar; d) un ciclu elementar.

8. Se dă un graf prin matricea sa de adiacenţă. Să se verifice dacă graful este arbore.

9. Se dă un graf cu n vîrfuri prin matricea sa de adiacenţă şi vîrfurile $v_1, v_2, \ldots, v_{n+1}$. Să se verifice dacă vîrfurile în ordinea dată formează un ciclu hamiltonian.

10. Se dă un graf cu n vîrfuri prin matricea sa de adiacenţă şi vîrfurile $v_1, v_2, \ldots, v_s$, unde $s > n$. Să se verifice dacă vîrfurile în ordinea dată formează un ciclu eulerian.

B

11. Se dă un graf prin matricea sa de adiacenţă şi un vîrf v. Să se elaboreze un algoritm care va parcurge graful prin metoda Breadth First, dacă vîrful de plecare este v.

12. Se dă un graf prin matricea sa de adiacenţă şi vîrfurile v_1, v_2. Să se verifice dacă există un lanţ care leagă vîrfurile v_1 şi v_2.

13. Se dă un graf prin matricea sa de adiacenţă. Să se afişeze la ecran lanţurile:
a) care conţin exact trei muchii ale grafului;
b) care conţin exact n muchii ale grafului, unde n este un număr natural dat.

14. Se dă un graf prin matricea sa de adiacenţă. Să se afişeze la ecran:
a) toate ciclurile grafului;
b) ciclurile care conţin exact trei muchii ale grafului;
c) ciclurile care conţin exact n muchii ale grafului, unde n este un număr natural dat.

15. Se dă un graf prin matricea sa de adiacenţă. Să se determine dacă graful este eulerian. În caz afirmativ, să se afişeze la ecran cel puţin un ciclu eulerian.

16. Se dă un graf prin matricea sa de adiacenţă. Să se determine dacă graful este hamiltonian. În caz afirmativ, să se afişeze la ecran cel puţin un ciclu hamiltonian.

17. Se dă un graf prin matricea sa de adiacenţă. Să se verifice dacă graful este arbore binar.

18. Se dă un arbore binar. Să se afişeze nodurile nivelului dat.

19. Preşedintele unei ţări este ales de Adunarea Generală din care fac parte n membri. Pentru a fi ales preşedintele trebuie să primească cel puţin $\frac{2}{3}$ din voturile membrilor. Între anumiţi membri ai Adunării Generale există conflicte de interese. Doi membri aflaţi în conflict de interese votează diferit. Fiind date numărul natural n şi perechile de numere (x, y), unde membrii cu numărul de ordine x şi y au conflict de interese, să se verifice dacă este posibilă alegerea preşedintelui.

20. Se dă un graf prin matricea sa de adiacenţă şi vîrfurile v_1, v_2. Să se determine lanţul (dacă există) de lungime minimă (care conţine cele mai puţine muchii) care leagă vîrfurile v_1 şi v_2.

21. Se dă un graf prin matricea sa de adiacenţă. Să se elaboreze un algoritm care adaugă (dacă este nevoie) numărul minim de muchii, astfel încît graful să devină conex. La ecran se vor afişa muchiile adăugate.

22. Se dau numerele naturale $d_1, d_2, ..., d_n$. Să se determine dacă există un arbore binar, ale cărui noduri au gradele, respectiv, $d_1, d_2, ..., d_n$.

23.* *Problema colorării hărţii.* Se consideră o hartă cu n ţări, $n > 3$. Pentru fiecare două ţări se ştie dacă ele sînt sau nu vecine. Avînd la dispoziţie doar 4 culori, să se coloreze harta, astfel încît fiecare două ţări vecine să fie colorate în culori diferite.

24. (*Olimpiada Naţională de Informatică, Bacău, România, 2001*)
La o competiţie au participat n concurenţi. Fiecare dintre ei a primit un număr de concurs de la 1 la n, astfel încît să nu existe concurenţi cu acelaşi număr. Din păcate, clasamentul final a fost pierdut, iar comisia îşi poate aduce aminte doar cîteva relaţii între unii participanţi (de genul „participatul cu numărul 3 a ieşit înaintea celui cu numărul 5"). Fiind date numărul natural n şi perechile ordonate de numere (x, y), unde

concurentul x a fost în clasament înaintea concurentului y, să se determine primul clasament în ordine lexicografică ce respectă relațiile pe care și le amintește comisia.

Problemele 25–31 pot fi rezolvate fără calculator

25. *Problema celor 7 poduri din Königsberg* (publicată în anul 1736 de Leonard Euler)

Orașul Königsberg (actualul Kaliningrad) se întinde pe ambele maluri ale rîului Pregel și pe două insule. Cele patru regiuni (A, B, C, D) ale orașului erau conectate în anul 1736 prin 7 poduri (fig. 6).

Se poate oare parcurge (făcind o plimbare) toate cele 7 poduri exact o singură dată?

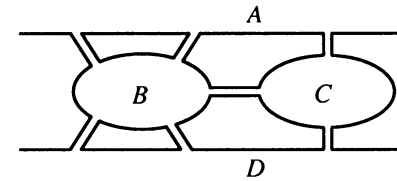


Fig. 6

26. În figura 7 o curbă neînchisă intersectează fiecare latură (segment) al pentagonului exact o singură dată.

Există oare o astfel de curbă pentru fiecare dintre următoarele figuri?

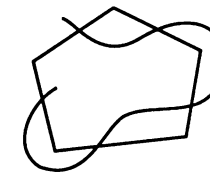


Fig. 7

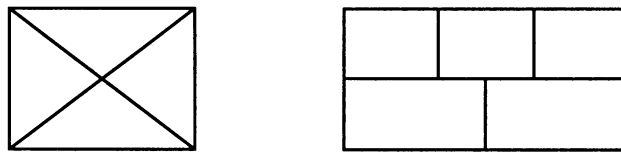

a) b)

Fig. 8

27. Se poate oare construi fiecare dintre următoarele figuri fără a ridica creionul și fără a desena un segment de 2 ori?

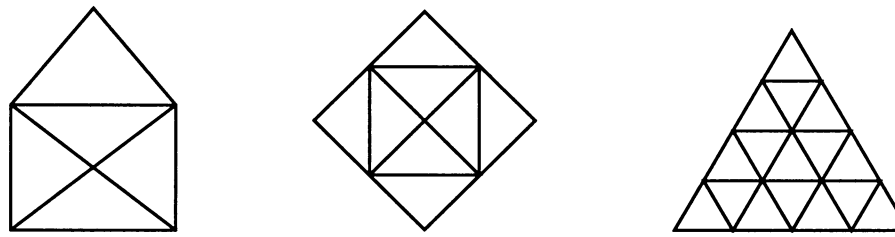

a) b) c)

Fig. 9

28. *Jocul icosian*

Inventat de Hamilton[1] în anul 1856, acest joc avea ca suport material un dodecaedru din lemn care reprezenta globul pămîntesc.

Vîrfurile dodecaedrului (fig. 10) erau notate cu numele unor orașe. Se cere să se găsească un drum închis pe muchiile dodecaedrului („voiaj în jurul lumii") care să parcurgă exact o singură dată fiecare vîrf.

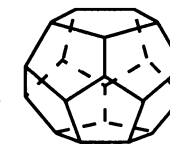


Fig. 10

[1] sir William Rowan Hamilton (1788–1856) – matematician, filosof și logician englez.

Indicaţie. Asociem dodecaedrului graful din figura 11, unde nodurile şi muchiile grafului corespund respectiv vîrfurilor şi muchiilor dodecaedrului.
Problema se reduce la construirea unui ciclu hamiltonian pentru graful din figura 11.

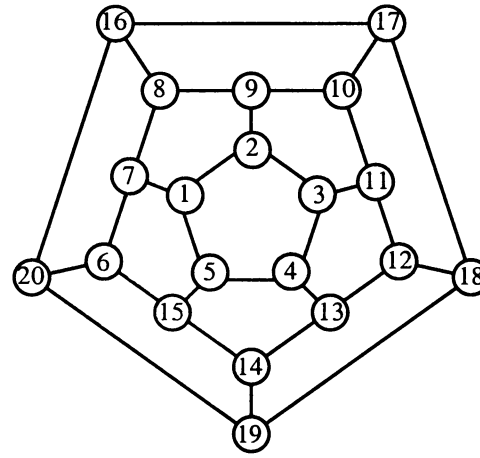


Fig. 11

29. Să se coloreze cele 12 regiuni din figura 12 cu 4 culori, astfel încît fiecare două regiuni cu frontiera comună să fie colorate cu diferite culori.

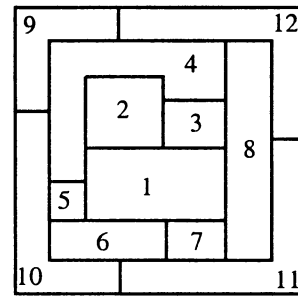


Fig. 12

30. (*Olimpiada de matematică a Republicii Moldova, 2005*)
Să se scrie toate cifrele de la 1 la 9 în rînd astfel, încît fiecare două cifre vecine să formeze un număr divizibil cu 7 sau cu 13.

31. [*Ivaşc C., Prună M.*, 4] La curtea regelui Artur s-au adunat $2n$ cavaleri şi fiecare din ei are printre cei prezenţi cel mult $n-1$ duşmani. Să se arate că Merlin, consilierul lui Artur, poate să-i aşeze în aşa fel pe cavaleri la o masă rotundă încît nici unul dintre ei să nu stea alături de vre-un duşman al său.
Indicaţie. Situaţia se va reprezenta printr-un graf cu $2n$ vîrfuri.
Se utilizează teorema lui Dirac[1] (1952): *Dacă* $G = (V, M)$ *este un graf cu* $n \geq 3$ *vîrfuri astfel încît fiecare vîrf* $v \in V$ *satisface condiţia* $d(v) \geq \frac{n}{2}$, *atunci* G *este graf hamiltonian.*

[1] Paul Adrien Maurice Dirac (n. 1902) – matematician şi fizician englez.

Anexa 1. Unit-ul CRT

Unit-ul Crt conține subprograme pentru:
- gestionarea regimului textual al monitorului;
- gestionarea generatorului de sunete;
- citirea tastaturii.

Regimul textual folosește următorul sistem de coordonate: vîrful stînga-sus al ecranului are coordonatele (0, 0).

Punctul de coordonate (x, y) este punctul obținut la intersecția liniei $x + 1$ și coloanei $y + 1$.

Variabile

`CheckBreak: (Boolean)` – Permite/interzice controlul Ctrl+Break
`CheckEOF: (Boolean)` – Permite/interzice controlul Ctrl+z
`DirectVideo: (Boolean)` – Permite/interzice accesul direct către memoria video
`CheckSnow: (Boolean)` – Permite/interzice controlul „fulgilor"
`LastMode: (Word)` – Păstrează regimul textual curent
`TextAttr: (Byte)` – Păstrează octetul curent al atributelor
`WindMin: (Word)` – Păstrează coordonatele vîrfului stînga-sus al ferestrei curente
`WindMax: (Word)` – Păstrează coordonatele vîrfului dreapta-jos al ferestrei curente

Constante

❶ Constantele regimului de lucru

`BW40` = `0` – Negru-alb, 40 de simboluri, 25 de linii
`BW80` = `2` – Negru-alb, 80×25
`Mono` = `7` – Monohrom, 80×25
`CO40` = `1` – Color, 40×25
`CO80` = `3` – Color, 80×25
`Font8x8` = `256` – Pentru regiumul EGA/VGA cu 43 sau 50 de linii
`C40` = `CO40` – Pentru compatibilitate cu versiunea 3.0
`C80` = `CO80` – Pentru compatibilitate cu versiunea 3.0

❷ Contantele culorilor

`Black`	= 0 – Negru
`Blue`	= 1 – Albastru
`Green`	= 2 – Verde
`Cyan`	= 3 – Cyan
`Red`	= 4 – Roşu
`Magenta`	= 5 – Violet
`Brown`	= 6 – Maro
`LightGray`	= 7 – Gri-deschis
`DarkGray`	= 8 – Gri-închis
`LightBlue`	= 9 – Albastru-aprins
`LightGreen`	= 10 – Verde-aprins
`LightCyan`	= 11 – Cyan-aprins
`LightRed`	= 12 – Roz
`LightMagenta`	= 13 – Magenta-aprins (zmeuriu)
`Yellow`	= 14 – Galben
`White`	= 15 – Alb
`Blink`	= 128 – stabileşte tipul de afişare intermitent (pulsarea simbolurilor)

Observaţii

1. Culorile 0, 1, ..., 7 pot fi folosite atît pentru scriere, cît şi pentru fundal, iar culorile 8, 9, ..., 15 pot fi folosite numai pentru scriere.

2. Constanta `Blink` poate însoţi una din cele 16 culori pentru a afişa intermitent.

Proceduri

1. **`AssignCrt`**`(var f:text)` – asociază fişierul text *f* cu fereastra (dispozitivul) CRT (CON – tastatura pentru citire, ecranul pentru afişare).

2. **`ClrEol`** – şterge toate caracterele din poziţia curentă a cursorului pînă la sfîşitul liniei curente.

3. **`ClrScr`** – şterge ecranul şi poziţionează cursorul în colţul stînga-sus.

4. **`GotoXY`**`(X,Y:byte)` – poziţionează cursorul în puctul de coordonate `(X,Y)`.

5. **`InsLine`** – inserează o linie goală în poziţia cursorului.

6. **`DelLine`** – şterge linia în care se află cursorul.

7. **`Delay`**`(MS:word)` – întrerupe executarea programului pentru `MS` milisecunde.

8. **`Sound`**`(Hz:word)` – conectează generatorul de sunete, generînd un sunet de frecvenţa `Hz` herţi.

9. **`NoSound`** – deconectează generatorul de sunete.

10. **`LowVideo`** – stabileşte o intensitate joasă a caracterelor.

11. **`NormVideo`** – restabileşte intensitatea normală a caracterelor.

12. **`TextBackGround`**`(C:byte)` – stabileşte culoarea `C` pentru fundal.

13. **`TextColor`**`(C:byte)` – stabileşte culoarea `C` pentru caractere.

14. **`TextMode`**`(Mode:word)` – stabileşte regimul textual `Mode`.

15. **`Window`**`(X1,Y1,X2,Y2:byte)` – stabileşte o fereastră de ecran, unde `(X1,Y1)` sînt coordonatele vîrfului stînga-sus, iar `(X2,Y2)` – coordonatele vîrfului dreapta-jos.

Funcţii

1. **`KeyPressed`**`:boolean` – returnează *True* dacă a fost apăsată o tastă după ultima citire din CRT. Nu întrerupe executarea programului.

2. **`ReadKey`**`:char` – citeşte un caracter de la tastatură. Caracterul citit nu este afişat la ecran.

3. **`WhereX`**`:byte` – returnează coordonata orizontalei a poziţiei curente a cursorului relativ de fereastra curentă.

4. **`WhereY`**`:byte` – returnează coordonata verticalei a poziţiei curente a cursorului relativ de fereastra curentă.

Anexa 2. Unit-ul Graph

Unit-ul Graph reprezintă o bibliotecă de subprograme pentru lucrul în regim grafic.

Pentru a lansa în execuţie un program care utilizează unit-ul Graph, este necesar un (sau mai multe) driver grafic (fişier cu extinderea .BGI, care va însoţi fişierul executabil .EXE). În cazul în care programul utilizează fonturi de haşurare, mai sînt necesare şi fişiere-fonturi (fişiere cu extinderea .CHR). Înainte de a face apel la un subprogram din Graph, regimul grafic trebuie iniţializat cu ajutorul procedurii `InitGraph`.

Constante

❶ Constantele driver-elor grafice

`CurrentDriver`	=	–128	– Pentru `GetModeRange`
`Detect`	=	0	– Autodetectarea
`CGA`	=	1	
`MCGA`	=	2	
`EGA`	=	3	
`EGA64`	=	4	
`EGAMono`	=	5	
`IBM8514`	=	6	
`HercMono`	=	7	
`ATT400`	=	8	
`VGA`	=	9	
`PC3270`	=	10	

❷ Regimuri (moduri) grafice pentru fiecare driver

`CGAC0`	=	0	=	320×200
`CGAC1`	=	1	=	320×200
`CGAC2`	=	2	=	320×200
`CGAC3`	=	3	=	320×200
`CGAHi`	=	4	=	640×200
`MCGAC0`	=	0	=	320×200
`MCGAC1`	=	1	=	320×200
`MCGAC2`	=	2	=	320×200
`MCGAC3`	=	3	=	320×200

MCGAMed	=	4	=	640×200
MCGAHi	=	5	=	640×480
EGAMonoHi	=	3	=	640×350
HercMonoHi	=	0	=	720×348
VGALo	=	0	=	640×200
VGAMed	=	1	=	640×350
EGALo	=	0	=	640×200
EGAHi	=	1	=	640×200
EGA64Lo	=	0	=	640×200
EGA64Hi	=	1	=	640×200
ATT400C0	=	0	=	320×200
ATT400C1	=	1	=	320×200
ATT400C2	=	2	=	320×200
ATT400C3	=	3	=	320×200
ATT400Med	=	4	=	640×200
ATT400Hi	=	5	=	640×400
IBM8514Lo	=	0	=	640×480
IBM8514Hi	=	1	=	1024×768
PC3270Hi	=	0	=	720×350
VGAHi	=	2	=	640×480

❸ Constantele returnate de `GraphResult`

grOk	=	0	–	Nu sînt greşeli.
grNoInitGraph	=	1	–	BGI driver-ul nu este instalat.
grNotDetected	=	2	–	Hardware-ul grafic nu este găsit.
grFileNotFound	=	3	–	Fişierul driver-ului grafic nu este găsit.
grInvalidDriver	=	4	–	Fişier incorect al driver-ului grafic.
grNoLoadMem	=	5	–	Nu este suficientă memorie pentru a încărca driver-ul.
grNoScanMem	=	6	–	Insuficientă memorie pentru afişarea domeniilor.
grNoFloodMem	=	7	–	Insuficientă memorie pentru colorarea domeniilor.
grFontNotFound	=	8	–	Fişierul fontului nu este găsit.
grNoFontMem	=	9	–	Insuficientă memorie pentru încărcarea fontului.
grInvalidMode	=	10	–	Mod grafic incorect pentru driver-ul ales.
grError	=	11	–	Eroare grafică (generală).
grIOerror	=	12	–	Eroare grafică de citire/extragere.
grInvalidFont	=	13	–	Fişier icorect al fontului.
grInvalidFontNum	=	14	–	Număr incorect al fontului.

❹ Constantele culorilor

`Black`	= 0	–	Negru
`Blue`	= 1	–	Albastru
`Green`	= 2	–	Verde
`Cyan`	= 3	–	Cyan
`Red`	= 4	–	Roşu
`Magenta`	= 5	–	Violet
`Brown`	= 6	–	Maro
`LightGray`	= 7	–	Gri-deschis
`DarkGray`	= 8	–	Gri-închis
`LightBlue`	= 9	–	Albastru-aprins
`LightGreen`	= 10	–	Verde-aprins
`LightCyan`	= 11	–	Cyan-aprins
`LightRed`	= 12	–	Roz
`LightMagenta`	= 13	–	Magenta-aprins (zmeuriu)
`Yellow`	= 14	–	Galben
`White`	= 15	–	Alb

❺ Constantele tipurilor liniilor (pentru `SetLineStyle`)

`SolidLn`	= 0	–	Continuă
`DottedLn`	= 1	–	Punctată
`CenterLn`	= 2	–	Ştrihpunctată
`DashedLn`	= 3	–	Întreruptă
`UserBitLn`	= 4	–	Tip determinat de utilizator

❻ Constantele grosimilor liniilor (pentru `SetLineStyle`)

`NormWidth`	= 1	–	Grosime normală
`ThickWidth`	= 3	–	Grosime triplă

❼ Constantele gestionării fonturilor

`DefaultFont`	= 0	–	Font bit mapped 8×8 (Matriceal)
`TriplexFont`	= 1	–	Font triplex (fişierul *trip.chr*)
`SmallFont`	= 2	–	Font mic (fişierul *litt.chr*)
`SansSerifFont`	= 3	–	Font drept (fişierul *sans.chr*)
`GothicFont`	= 4	–	Font gotic (fişierul *goth.chr*)
`HorizDir`	= 0	–	Direcţie orizontală
`VertDir`	= 1	–	Direcţie verticală
`UserCharSize`	= 0	–	Mărimea simbolurilor este determinată de utilizator

Următoarele constante se utilizează cu procedura `SetTextJustify`

`LeftText`	= 0	–	Alinierea la stînga
`CenterText`	= 1	–	Alinierea la centru (pe orizontală)
`RightText`	= 2	–	Alinierea la dreapta

`BottomText`	=	0	–	Alinierea jos
`CenterText`	=	1	–	Alinierea la centru (pe verticală)
`TopText`	=	2	–	Alinierea sus

❽ Constantele şabloanelor de haşurare

Notă. Se utilizează ca parametri ai procedurilor `GetFillSettings` şi `SetFillStyle`.

`EmptyFill`	=	0		Fără haşurare
`SolidFill`	=	1	-	Haşurare continuă (neîntreruptă)
`LineFill`	=	2	—	Haşurare
`LtSlashFill`	=	3	///	Haşurare
`SlashFill`	=	4	///	Haşurare groasă {thick}
`BkSlashFill`	=	5	\\\	Haşurare {thick}
`LtBkSlashFill`	=	6	\\\	Fill haşurare
`HatchFill`	=	7	+++	Haşurare {Light hatch fill}
`XHatchFill`	=	8	xxx	Haşurare {Heavy cross hatch}
`InterleaveFill`	=	9		Haşurare sub formă de pătrăţele {Interleaving line}
`WideDotFill`	=	10		Haşurare cu puncte, rară {Widely spaced dot}
`CloseDotFill`	=	11		Haşurare cu puncte, densă {Closely spaced dot}
`UserFill`	=	12		Haşurarea se defineşte de utilizator

❾ Constantele pentru procedura `PutImage`

`NormalPut`	=	0	–	`MOV`
`CopyPu`	=	0	–	`MOV`
`XORPut`	=	1	–	`XOR`
`OrPu`	=	2	–	`OR`
`AndPut`	=	3	–	`AND`
`NotPut`	=	4	–	`NOT`

❿ Alte constante

{Următoarele două constante se utilizează în procedura `SetViewPort`}

`ClipOn`	=	*True*	–	Imaginea va fi trunchiată (va fi acoperită de fereastra curentă).
`ClipOff`	=	*False*	–	Imaginea nu va fi trunchiată.

{Următoarele două constante se utilizează în procedura `Bar3D`}

`TopOn`	=	*True*	–	Se desenează vîrful paralelipipedului.
`TopOff`	=	*False*	–	Nu se desenează vîrful paralelipipedului.
`MaxColors`	=	15	–	Numărul maximal de culori.

Tipuri

```
PaletteType=record {Se utilizeaza in procedura GetPallete}
   Size:Byte;
   Colors:array[0..MaxColors] of Shortint;
end;

LineSettingsType=record {Se utilizeaza in procedura GetLineSettings}
   LineStyle:Word;
   Pattern:Word;
   Thickness:Word;
end;

TextSettingsType=record {Se utilizeaza in procedura GetTextSettings}
   Font:Word;
   Direction:Word;
   CharSize:Word;
   Horiz:Word;
   Vert:Word;
end;

FillSettingsType=record {Se utilizeaza in procedura GetFillSettings}
   Pattern:Word;
   Color:Word;
end;

FillPatternType=array[1..8] of Byte;

PointType=record {Pentru a stabili coordonatele virfurilor poligoanelor}
   X,Y:integer;
end;

ViewPortType=record {Se utilizeaza in procedura GetViewSettings}
   x1,y1,x2,y2:integer;
   Clip:Boolean;
end;
```

Variabile

`GraphGetMemPtr:pointer` – Organizează Heap-ul unit-ului Graph.

`GraphFreeMemPtr:pointer` – Eliberează Heap-ul unit-ului Graph.

Notă. Se vor utiliza atunci cînd programul utilizatorului va conţine algoritmi proprii de gestionare a Heap-ului unit-ului Graph.

Proceduri

1. **Arc**`(X,Y;Integer;StAngle,EndAngle,Radius;Word)` – desenează un arc de cerc de rază `Radius` şi centru `(X,Y)` cu extremităţile `StAngle`, `EndAngle` (exprimate în grade).
2. **Bar**`(x1,y1,x2,y2:Integer)` – desenează o fişie dreptunghiulară, unde `(x1,y1)` şi `(x2,y2)` sînt respectiv coordonatele vîrfurilor stînga-sus şi dreapta-jos (utilizează stilul şi culoarea curentă).

3. **Bar3D**(x1,y1,x2,y2:Integer;Depth:Word;Top:Boolean) – desenează o fişie dreptunghiulară tridimensională (paralelipiped), unde (x1,y1) şi (x2,y2) sînt respectiv coordonatele vîrfurilor stînga-sus şi dreapta-jos (utilizează stilul şi culoarea curentă). Dacă Top este *true*, atunci se desenează vîrful paralelipipedului.
4. **Circle**(X,Y:Integer;Radius:Word) – desenează un cerc de centru (X,Y) şi rază Radius.
5. **ClearDevice** – şterge ecranul.
6. **ClearViewPort** – şterge fereastra.
7. **CloseGraph** – închide regimul grafic şi restabileşte regimul textual al ecranului (memoria ocupată de driver-ul grafic se eliberează).
8. **DetectGraph**(**var** GraphDriver,GraphMode:Integer) – returnează tipul driver-ului instalat şi regimul de lucru al lui.
9. **DrawPoly**(NumPoints:Word; **var** PolyPoints) – desenează un poligon cu NumPoints vîrfuri cu coordonatele PolyPoints (utilizează culoarea şi tipul liniei curente).
10. **Ellipse**(X,Y:Integer;StAngle,EndAngle:Word;XRadius,YRadius:Word) – desenează un arc de elipsă de centru (X,Y), extremităţile StAngle, EndAngle şi razele orizontale şi verticale respectiv XRadius, YRadius.
11. **FillEllipse**(X,Y:Integer;XRadius,YRadius:Word) – desenează o elipsă haşurată de centru (X,Y) şi raze XRadius, YRadius.
12. **FillPoly**(NumPoints:Word; **var** PolyPoints) – desenează şi haşurează un poligon cu NumPoint vîrfuri şi coordonatele PolyPoints.
13. **FloodFill**(X,Y:Integer;Border:Word) – haşurează un domeniu închis ce conţine punctul (X,Y) şi este mărginit de linia de culoare Border (utilizează stilul de haşurare şi culoarea curentă).
14. **GetArcCoords**(**var** ArcCoords:ArcCoordsType) – returnează coordonatele şi extremităţile arcului de cerc.
15. **GetAspectRatio**(**var** Xasp,Yasp:Word) – returnează două numere prin care se poate determina raportul dimensiunilor ecranului.
16. **GetDefaultPalette**(**var** Palette:PaletteType) – returnează paleta curentă.
17. **GetFillPattern**(**var** FillPattern:FillPatternType) – returnează modul curent de haşurare.
18. **GetFillSettings**(**var** FillInfo:FillSettingsType) – returnează modul şi culoarea de haşurare curentă.
19. **GetImage**(x1,y1,x2,y2:Integer; **var** BitMap) – păstrează în variabila BitMap fragmentul dreptunghiular de ecran, ale cărui vîrfuri stînga-sus şi dreapta-jos au coordonatele (x1,y1) şi (x2,y2).
20. **GetLineSettings**(**var** LineInfo:LineSettingsType) – returnează stilul, şablonul şi grosimea curente ale liniei.
21. **GetModeRange**(GraphDriver:Integer; **var** LoMode,HiMode:Integer) – returnează pentru driver-ul grafic indicat diapazonul posibil al regimurilor de lucru.

22. **GetPalette**(**var** Palette:PaletteType) – returnează paleta curentă şi dimensiunile ei.
23. **GetTextSettings**(**var** TextInfo:TextSettingsType) – returnează opţiunile (fontul, direcţia, dimensiunea şi alinierea) curente ale textului, stabilite cu procedurile SetTextStyle şi SetTextJustify.
24. **GetViewSettings**(**var** ViewPort:ViewPortType) – returnează coordonatele şi informaţia despre trunchiere a ferestrei curente.
25. **GraphDefaults** – stabileşte parametrii standard ai regimului grafic.
26. **InitGraph**(**var** Gd:Integer; **var** Gm:Integer;PathToDriver:string) – iniţializează regimul grafic, unde Gd este tipul driver-ului grafic, Gm – regimul lui, iar PathToDriver reprezintă calea pînă la driver-ul grafic. Gd poate fi 0, dacă se doreşte determinarea automată a tipului şi regimului.
27. **Line**(x1,y1,x2,y2:Integer) – desenează un segment ale cărui extremităţi sînt punctele de coordonate (x1,y1) şi (x2,y2).
28. **LineRel**(Dx,Dy:Integer) – desenează un segment, ale cărui extremităţi sînt coordonatele indicatorului (IndX,IndY) şi punctul de coordonate (IndX+Dx, IndY+Dy).
29. **LineTo**(X,Y:Integer) – desenează un segment, ale cărui extremităţi sînt coordonatele indicatorului (IndX,IndY) şi punctul de coordonate (X,Y).
30. **MoveRel**(Dx,Dy:Integer) – deplasează indicatorul curent din punctul iniţial (IndX, IndY) în punctul (IndX+Dx,IndY+Dy).
31. **MoveTo**(X,Y:Integer) – deplasează indicatorul curent în punctual de coordonate (X,Y).
32. **OutText**(TextString:string) – afişează la ecran textul TextString în poziţia indicatorului.
33. **OutTextXY**(X,Y:Integer;TextString:string) – afişează la ecran textul TextString în poziţia (X,Y).
34. **PieSlice**(X,Y:Integer;StAngle,EndAngle,Radius:Word) – desenează şi haşurează un sector de cerc de rază Radius şi centru (X,Y) cu extremităţile StAngle, EndAngle (exprimate în grade).
35. **PutImage**(X,Y:Integer; **var** BitMap;BitBlt:Word) – plasează fragmentul dreptunghiular de ecran memorizat anterior cu PutImage în BitMap. Colţul stînga-sus va avea coordonatele (X,Y).
36. **PutPixel**(X,Y:Integer;Color:Word) – desenează punctul de coordonate (X,Y) şi culoare Color.
37. **Rectangle**(x1,y1,x2,y2:Integer) – desenează un dreptunghi a cărui diagonală este segmentul cu extremităţile de coordonate (x1,y1), (x2,y2).
38. **RestoreCrtMode** – restabileşte regimul textual al ecranului, neeliberînd memoria ocupată de driver-ul grafic.
39. **Sector**(x,y:Integer;StAngle,EndAngle,XRadius,YRadius:Word) – desenează şi haşurează un sector de cerc de raze XRadius, YRadius şi centru (X,Y) cu extremităţile StAngle, EndAngle (exprimate în grade).

40. **SetActivePage** (Page:Word) – stabileşte pagina activă pentru afişări.
41. **SetAllPalette** (**var** Palette) – modifică culorile paletei.
42. **SetAspectRatio** (Xasp,Yasp:Word) – modifică coeficientul de proporţionalitate a dimensiunilor ecranului.
43. **SetBkColor** (Color:Word) – stabileşte culoarea pentru fundal.
44. **SetColor** (Color:Word) – stabileşte culoarea de bază pentru desenare.
45. **SetFillPattern** (Pattern:FillPatternType;Color:Word) – stabileşte modul de haşurare definit de utilizator.
46. **SetFillStyle** (Pattern:Word;Color:Word) – stabileşte modul de haşurare şi culoarea.
47. **SetGraphBufSize** (BufSize:Word) – modifică dimensiunile bufferului pentru funcţii de haşurare.
48. **SetGraphMode** (Mode:Integer) – curăţă ecranul şi stabileşte un alt regim grafic.
49. **SetLineStyle** (LineStyle:Word;Pattern:Word;Thickness:Word) – stabileşte grosimea şi tipul liniei.
50. **SetPalette** (ColorNum:Word;Color:Shortint) – modifică culoarea paletei cu numărul ColorNum prin culoarea Color.
51. **SetRGBPalette** (ColorNum,RedValue,GreenValue,BlueValue:Integer) – modifică paleta pentru IBM 8514 şi drivere VGA.
52. **SetTextJustify** (Horiz,Vert:Word) – stabileşte alinierea textului, utilizată în procedurile OutText şi OutTextXY.
53. **SetTextStyle** (Font,Direction:Word;CharSize:Word) – stabileşte fontul, stilul şi dimensiunea textului.
54. **SetUserCharSize** (MultX,DivX,MultY,DivY:Word) – modifică proporţiile fontului.
55. **SetViewPort** (x1,y1,x2,y2:Integer;Clip:Boolean) – stabileşte fereastra curentă pentru afişări grafice.
56. **SetVisualPage** (Page:Word) – stabileşte numărul paginii grafice afişate.
57. **SetWriteMode** (WriteMode:Integer) – stabileşte regimul de afişare (copiere sau XOR) pentru linii (desenate cu DrawPoly, Line, LineRel, LineTo, Rectangle).

Funcţii

1. **GetBkColor**:Word – returnează culoarea curentă a fundalului.
2. **GetColor**:Word – returnează culoarea curentă pentru desenare.
3. **GetDriverName**:String – returnează numele driver-ului curent.
4. **GetGraphMode**:Integer – returnează numărul regimului grafic curent.
5. **GetMaxColor**:Word – returnează numărul maximal (corespunzător unei culori) care poate fi indicat în SetColor.
6. **GetMaxMode**:Integer – returnează numărul maximal corespunzător unui regim al driver-ului curent.

7. **GetMaxX**: `Integer` – returnează coordonata maximală a orizontalei ecranului (în regim grafic).
8. **GetMaxY**: `Integer` – returnează coordonata maximală a verticalei ecranului (în regim grafic).
9. **GetModeName** `(ModeNumber:Integer):string` – returnează denumirea regimului grafic `ModeNumber`.
10. **GetPaletteSize**: `Integer` – returnează volumul tabelei paletei.
11. **GetPixel** `(X,Y:Integer):Word` – returnează culoarea punctului de coordonate `(X,Y)`.
12. **GetX**: `Integer` – returnează coordonata orizontalei indicatorului poziţiei curente.
13. **GetY**: `Integer` – returnează coordonata verticalei indicatorului poziţiei curente.
14. **GraphErrorMsg** `(ErrorCode:Integer):string` – returnează mesajul corespunzător valorii `ErrorCode`, returnată de `GraphResult`.
15. **GraphResult**: `Integer` – returnează codul rezultatului ultimei adresări către o procedură grafică.
16. **ImageSize** `(x1,y1,x2,y2:Integer):Word` – returnează numărul de octeţi necesari pentru păstrarea în memorie a fragmentului dreptunghiular de ecran, ale cărui vîrfuri stînga-sus şi dreapta-jos au coordonatele `(x1,y1)` şi `(x2,y2)`.
17. **InstallUserDriver** `(Name:string;AutoDetectPtr:pointer):integer` – instalează driver-ul utilizatorului în tabela driver-elor. Dacă tabela este completă se returnează valoarea –11 (`grError`), altfel – numărul din tabelă corespunzător driver-ului instalat.
18. **InstallUserFont** `(FontFileName:string):Integer` – instalează un font nou, care nu există în sistemul BGI.
19. **RegisterBGIdriver** `(driver:pointer):Integer` – înregistrează driver-ul pentru sistemul grafic. Dacă este depistată o greşeală, se returnează o valoare mai mică decît 0, altfel – numărul corespunzător driver-ului.
20. **RegisterBGIfont** `(Font:pointer):Integer` – înregistrează fontul pentru sistemul grafic. Dacă este depistată o greşeală, se returnează o valoare mai mică decît 0, altfel – numărul corespunzător fontului.
21. **TextHeight** `(TextString:string):Word` – returnează înălţimea (numărul de pixeli) textului `TextString`.
22. **TextWidth** `(TextString:string):Word` – returnează lăţimea (numărul de pixeli) textului `TextString`.

Anexa 3. Unit-ul DOS

Unit-ul Dos conţine subprograme pentru gestionarea fişierelor şi crearea programelor pentru sistemul de operare MS DOS.

Constante

❶ Constantele registrului `Flags`

```
fCarry      = $0001
fParity     = $0004
fAuxiliary  = $0010
fZero       = $0040
fSign       = $0080
fOverflow   = $0800
```

❷ Constantele modului de acces către fişier

```
fmClosed    = $D7B0 {fişier închis}
fmInput     = $D7B1 {fişier deschis doar pentru citire}
fmOutput    = $D7B2 {fişier deschis doar pentru scriere}
fmInOut     = $D7B3 {fişier deschis pentru citire şi scriere}
```

❸ Constantele atributelor fişierelor

```
ReadOnly    = $01 {fişier doar pentru citire}
Hidden      = $02 {fişier ascuns}
SysFile     = $04 {fişier de sistem}
VolumeID    = $08 {etichetă}
Directory   = $10 {catalog}
Archive     = $20 {fişier de arhivă}
AnyFile     = $3F {orice fişier}
```

Tipuri

```
{fisiere cu tip si fara tip}
FileRec=record
   Handle:Word;
   Mode:Word;
   RecSize:Word;
```

```
   Private:array[1..26] of Byte;
   UserData:array[1..16] of Byte;
   Name:array[0..79] of Char;
end;

{fisiere text}
TTextBuf=array[0..127] of Char;
   TTextRec=record
      Handle:Word;
      Mode:Word;
      BufSize:Word;
      Private:Word;
      BufPos:Word;
      BufEnd:Word;
      BufPtr:PTextBuf;
      OpenFunc:Pointer;
      InOutFunc:Pointer;
      FlushFunc:Pointer;
      CloseFunc:Pointer;
      UserData:array[1..16] of Byte;
      Name:array[0..79] of Char;
      Buffer:TTextBuf;
   end;

Registers=record
   case Integer of
   0:(AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Word);
   1:(AL,AH,BL,BH,CL,CH,DL,DH:Byte);
end;

DateTime=record
   Year,Month,Day,Hour,Min,Sec:Word;
end;

SearchRec=record
   Fill:array[1..21] of Byte;
   Attr:Byte;
   Time:Longint;
   Size:Longint;
   Name:string[12];
end;

DirStr     =   string[67] {nume disc sau catalog}
NameStr    =   string[8] {nume fisier}
ExtStr     =   string[4] {extindere fisier}
ComStr     =   string[127] {linie de comanda}
PathStr    =   string[79] {calea completa de cautare a fisierului}
```

Variabile

DosError: Integer; {memorează codul erorii sistemului de operare}

Aceste coduri sînt:

0 – nu este eroare.

2 – fişierul nu există.

3 – calea nu este găsită.
5 – accesul este interzis.
6 – prelucrare incorectă.
8 – insuficientă memorie.
10 – mediu incorect.
11 – format incorect.
18 – alte fişiere nu există.

Proceduri

1. **GetDate**(**var** Year,Month,Day,DayOfWeek:Word) – returnează data curentă. Domeniul de valori al lui Year este 1980...2099, al lui Month este 1...12, al lui Day este 1...31, iar al lui DayOfWeek este 0...6 (unde 0 corespunde sîmbetei).
2. **GetFTime**(**var** F; **var** Time:Longint) – returnează data şi timpul ultimei modificări în fişier.
3. **GetTime**(**var** Hour,Minute,Second,Sec100:Word) – returnează timpul curent (al sistemului de operare).
4. **PackTime**(**var** T:DateTime; **var** Time:Longint) – transformă înscrierea T într-o valoare a datei şi timpului de 4 octeţi (tip LongInt), valoare utilizată de procedura SetFTime. Domeniile înscrierii T nu se verifică.
5. **SetDate**(Year,Month,Day:Word) – stabileşte data curentă.
6. **SetFTime**(**var** F;Time:Longint) – stabileşte timpul şi data ultimei modificări în fişier.
7. **SetTime**(Hour,Minute,Second,Sec100:Word) – stabileşte timpul curent.
8. **UnpackTime**(Time:Longint; **var** DT:TDateTime) – transformă valoarea datei şi timpului de 4 octeţi, returnată de GetFTime, FindFirst sau FindNext, într-o înscriere de tipul DateTime.
9. **GetIntVec**(IntNo:Byte; **var** Vector:Pointer) – returnează adresa păstrată în vectorul indicat al întreruperii.
10. **Intr**(IntNo:Byte; **var** Regs:TRegisters) – execută întreruperea de program indicată.
11. **MsDos**(**var** Regs:Registers) – execută o funcţie a sistemului de operare.
12. **SetIntVec**(IntNo:Byte;Vector:Pointer) – stabileşte adresa vectorului întreruperii indicat.
13. **FSplit**(Path:PathStr; **var** Dir:DirStr; **var** Name:NameStr; **var** Ext:ExtStr) – divizează numele fişierului Path în 3 părţi: Dir – catalogul, Name – numele, Ext – extinderea.
14. **FindFirst**(Path:PChar; Attr:Word; **var** F:TSearchRec) – caută în directoriul indicat sau în cel curent primul fişier, ale cărui nume şi atribute coincid cu cele indicate.
15. **FindNext**(**var** F:TSearchRec) – returnează următorul fişier cu numele şi atributele specificate în apelul anterior al procedurii FindFirst.

16. **GetFAttr** (**var** F; **var** Attr:Word) – returnează atributele fişierului F. (Variabila F este de tip fişier.)
17. **SetFAttr** (**var** F;Attr:Word) – stabileşte atributele fişierului F.
18. **Exec** (Name,CmdLine:string) – lansează în execuţie fişierul executabil Name cu linia de comandă CmdLine.
19. **Keep** (ExitCode:Word) – sfîrşeşte execuţia programului, acesta devenind rezident.
20. **SwapVectors** – vectorii întreruperii ai variabilelor SaveIntXX salvaţi sînt substituiţi cu vectorii curenţi.
21. **GetCBreak** (**var** Break:Boolean) – returnează starea Ctrl-Break.
22. **SetCBreak** (Break:Boolean) – stabileşte starea Ctrl-Break.
23. **GetVerify** (**var** Verify:Boolean) – returnează starea opţiunii de verificare în Dos.
24. **SetVerify** (Verify:Boolean) – stabileşte starea opţiunii de verificare în DOS.

Funcţii

1. **DiskFree** (Drive:Byte):Longint – returnează numărul de octeţi disponibili pe discul indicat, unde 0 este discul curent, 1 – discul A:, 2 – discul B:, 3 – discul C: etc. Dacă se returnează valoarea –1, atunci discul indicat nu este găsit (nu există).
2. **DiskSize** (Drive:Byte):Longint – returnează numărul total de octeţi ai discului indicat.
3. **FExpand** (Name:String):PathStr – returnează numele complet (împreună cu calea) al fişierului Name.
4. **FSearch** (Name:String; DirList:string):PathStr – caută fişierul Name în lista de cataloage DirList. Componentele listei se delimitează prin virgulă.
5. **DosExitCode**:Word – returnează codul de sfîrşit al subprocesului. Aceste coduri pot fi:

Termination Type	High Byte
Normal	0
Ctrl-C	1
Device error	2
Keep procedure	3

6. **EnvCount**:Integer – returnează numărul de string-uri (de forma Var = valoare, deci numărul de variabile) ale mediului DOS.
7. **EnvStr** (Index:Integer):string – returnează string-ul (de forma Var = valoare) indicat al mediului DOS.
8. **GetEnv** (EnvVar:string):string – returnează valoarea variabilei indicate a mediului DOS.
9. **DosVersion**:Word – returnează numărul versiunii DOS.

BIBLIOGRAFIE

1. Atanasiu A., Pintea R., *Culegere de probleme Pascal*, Editura „Petrion", Bucureşti, 1995.

2. Dogaru O., Petcu D., Petrov Gh., *Turbo Pascal. Exerciţii şi probleme*, Editura de vest, Timişoara, 1995.

3. Gremalschi A., Mocanu I., Spinei I., *Informatica. Limbajul Pascal. Manual pentru clasele IX–XI*, Î.E.P., „Ştiinţa", Chişinău, 1999.

4. Ivaşc C., Prună M., *Bazele Informaticii (Grafuri şi Elemente de Combinatorică). Proiect de manual pentru clasa a X-a. Profil Informatică*, Editura „Petrion", Bucureşti, 1995.

5. Ivaşc C., Prună M., *Tehnici de programare. Aplicaţii*, Editura „Petrion", Bucureşti, 1999.

6. Munteanu F., Ionescu T., Muscă Gh., Tătaru D., Dascălu S., *Programarea calculatoarelor. Manual pentru licee de informatică. Clasele X–XII*, Editura didactică şi pedagogică, R.A. – Bucureşti.

7. Niculescu S., Butnaru V., Vlad M., *Informatică. Manual pentru clasa a X-a (profilul matematică-informatică)*, Editura „Teora", Bucureşti, 2000.

8. Popescu Anastasiu D., *Culegere de probleme de informatică pentru gimnaziu şi liceu*, Editura „All Educational", Bucureşti, 2000.

9. Popovici P., *Structuri de date liniare şi arborescente*, Editura „Eubeea", Timişoara, 2002.

10. Sorin T., *Tehnici de programare. Manual pentru clasa a X-a*, „Editura L&S Infomat", Bucureşti, 1996.

11. Tomescu I., *Grafuri şi programare liniară. Introducere elementară*, Editura didactică şi pedagogică, Bucureşti, 1975.

12. Абрамов С. А., Гнездилова Г. Г., Капустина Е. Н., Селюн М. И., *Задачи по программированию*, Издательство „Наука", Москва, 1988.

13. Пильщиков В. Н, *Сборник упражнений по языку Паскаль*, Издательство „Наука", Москва, 1989.

14. Немнюгин С. С., *Turbo Pascal. Практикум*, Издательский дом „Питер", Санкт-Петерсбург, 2003.

15. Фаронов В. В., *Turbo Pascal. Начальный курс. Учебное пособие*, Издательство „Нолидж", Москва, 2001.